JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 14, No. 4, April 2017

Forward Adaptive Transfer of Gaussian Process Regression

Neeti Wagle* and Eric W. Frew†
University of Colorado, Boulder, Colorado 80309

DOI: 10.2514/1.I010437

This paper proposes a forward adaptive transfer learning method, referred tp as forward adaptive transfer learning for Gaussian process regression, which allows robots to leverage previously learned Gaussian process regression models and use them as sources of information in new learning tasks. This is especially valuable when limited training data are available for the new target task. Forward adaptive transfer learning for Gaussian process regression decouples the kernel and hyperparameter selection for the target task from those of the source task, providing an inference framework that is desirable when dealing with real-world dynamic environments. Finally, because the source task has been learned a priori, the computational complexity of forward adaptive transfer learning for Gaussian process regression is lower as compared to other semiparametric Gaussian process approaches using transfer kernels.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

Nomenclature

b	=	coefficient of transfer
$C_{(\cdot)}$	=	covariance matrix
c_{TS}, c_S, $c_{\tau(T)}$	=	parameters in definition of b
$f(\boldsymbol{x})$, $g(\boldsymbol{x})$	=	simulation example functions
$\mathcal{GP}_{(\cdot)}$	=	Gaussian process model
$K_{(\cdot)}$	=	component of covariance matrix determined from Gaussian process kernel functions
$k_{(\cdot)}(\boldsymbol{x}_i, \boldsymbol{x}_j, \theta_{(\cdot)})$	=	kernel function
l, L	=	scalar length scale and diagonal matrix of length scales
$\boldsymbol{m}_S$, $\boldsymbol{m}_T$, $\boldsymbol{m}_{\text{joint}}$	=	mean vectors for Gaussian process model
N_T, N_S	=	numbers of measurements
S, T	=	source and target task indicators
X	=	set of measurement states
$\boldsymbol{x}$	=	input variable
$\boldsymbol{x}'$, y'	=	unseen input variable and predicted response
Y	=	set of measurements
y	=	response variable
θ_S, θ_T, θ_τ	=	hyperparameters
λ	=	scaling factor indicating cross correlation
μ	=	Gaussian function mean
σ_n, σ_f	=	sensor noise and signal variance hyperparameters

I. Introduction

NONPARAMETRIC learning techniques provide a data-driven approach that is useful for robotic missions in unstructured, diverse, and unexplored environments. One such example is the use of Gaussian process (GP) regression [1,2] to capture various environmental phenomena. GPs accommodate diverse spatial and spatiotemporal behaviors using a single methodology. By capturing the inherent characteristics of the environment, a GP can provide fully probabilistic predictions at any location in the environment. Due to their flexibility and ability to handle uncertainty, GPs are becoming increasingly ubiquitous in robotic learning tasks. They have been used for monitoring environmental phenomena [3,4], mapping the terrain in which the robot navigates [5,6], modeling motion control and dynamics [7–9], as well as tracking dynamic obstacles [10]. They have also been popular in communication modeling [11,12] and energy harvesting [13,14] for robots in terrestrial, aerial, and aquatic environments.

To leverage the advantages of GPs fully, the learned models must be adaptable to subsequent robotic missions in these dynamic environments. Unfortunately, the GP's training is tightly coupled to specific environmental conditions, and changes to the environment in subsequent missions can significantly reduce the predictive power of the model. Furthermore, learning a GP is a computationally expensive process. The learning phase requires the inversion of a covariance matrix that is the size of the training dataset N, causing the learning process to M scale as $O(N^3)$ in the naive case or linearly using sparse approximation techniques [15,16]. Instead of repeating the expensive and extensive data collection and training phases, transfer learning can help reuse the relevant knowledge from the old model. In this way, the existing GP model can act as a source of information for updating the new target task GP model [17] using much less data, e.g., measurements with $M \ll N$. This flavor of transfer learning, wherein the source and target tasks are the same but deal with different data distributions, is known as transductive transfer learning or domain adaptation [18,19].

The motivating application for this work is learning spatiotemporal models of the radio-frequency environment and communication channel performance for communication-aware planning in robot sensor networks [11,20–25]. Gaussian processes have been used to capture the geospatial behavior of communication channels [24,26,27]. The GP provides predictions of communication performance at future locations,

Received 26 November 2015; revision received 4 October 2016; accepted for publication 8 December 2016; published online 6 April 2017. Copyright © 2016 by Neeti Wagle. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the ISSN 2327-3097 (online) to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Ph.D. Candidate, Department of Computer Science. Student Member AIAA.
†Associate Professor, Department of Aerospace Engineering Sciences. Associate Fellow AIAA.

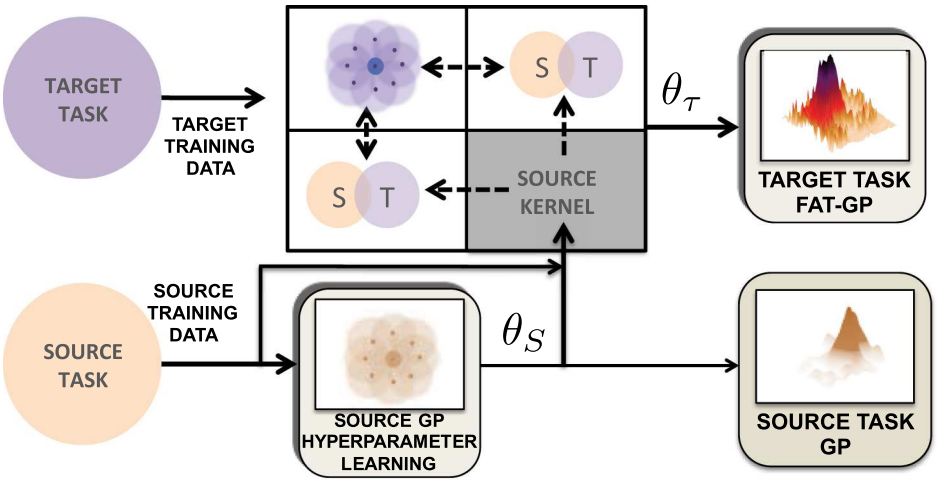


Fig. 1 Overview of the FAT-GP algorithm that uses the a priori source task GP to learn the new target task GP.

allowing for proactive planning by the robot team. Transfer learning can accelerate the learning process across multiple cooperating robots, allowing the robot team to shorten the exploration phase (learning the model) of a mission and lengthen the exploitation phase (applying the model). Scenarios where transfer learning could be helpful include learning models for a robot with different onboard radios deployed to the same environment as another robot; use of a new base station with new antenna gains in an environment that has been explored previously; deployment to an environment that has already been explored but where atmospheric conditions have changed; etc. Preliminary work by the authors explored simple transfer learning of communication models through hyperparameter transfer [17,28]. Based on flight-test data, those results demonstrated poor transfer performance and motivated the need for the domain adaptation approach presented here.

Most GP-based transductive transfer learning has focused on multitask learning, often viewing similar or related tasks as samples from the same Gaussian process [29–31]. More recently, semiparametric Gaussian process approaches have naturally extended the GP framework for domain adaptation from a source task to a target task. The adaptive transfer learning GP (AT-GP) is a prominent example of this approach [32,33]. The AT-GP uses a transfer kernel to learn task similarity, and it tunes the impact that the source task has on predictions for the target task. Unfortunately, the AT-GP constrains the source and target tasks to the same kernel (function and hyperparameters), resulting in an implicit transfer even when the tasks have zero or little correlation. In addition, the AT-GP formulation does not allow transfer from already learned source tasks to new target tasks, only providing a method for simultaneously learning the two tasks. When transfer learning is employed in robotic learning tasks with the objective of amortizing training costs, the additional computational complexity of learning the source's hyperparameters is especially undesirable.

In this paper, we present forward adaptive transfer learning for Gaussian process regression (FAT-GP), which allows previously learned GP models to be adapted forward as potential sources of knowledge for future learning tasks. FAT-GP combines the source task's previously learned model, the source task's training data, and the target task's training data to learn the target hyperparameters as well as the correlation between the two tasks (Fig. 1). As a result, the new FAT-GP is able to use both the target task and source task training data to predict outputs of the target task without needing to perform online the costly inversion of the large ($N \gg M$) source task covariance matrix.

FAT-GP decouples the kernel and hyperparameter selection for the target task from those of the source task. Because the source task reuses already learned hyperparameters, its large covariance matrix can be precomputed, reducing the computational complexity of the training. To ensure that the joint covariance matrix is positive semidefinite, the cross-covariance kernel function is defined as a product of the task similarity and a kernel function that averages over the length scales local to each of the tasks.

The forward adaptive Gaussian process is an example of the broad class of asymmetric learning algorithms where the learning transfer only happens in one direction: from the source to the target. In contrast, symmetric learning such as multitask learning also applies target task data back to the source task. Asymmetric transfer learning with Gaussian processes has been used for visual domain adaptation [34–36]. Those works focused on classification problems as opposed to the regression tasks of interest in this paper. A similar concept to the FAT-GP is focused multitask learning [37], where asymmetric transfer is enabled during multitask learning by assuming there exist a primary (source) task and secondary (target) tasks that can be modeled with covariance matrices that are the sum of two components: one that uses a kernel function with the same hyperparameters as the primary task, and one that has separate learned hyperparameters that can "explain away" [37] the differences between the tasks. The FAT-GP is not derived with the same underlying assumption, but it leads to a source task covariance with similar form. Furthermore, the FAT-GP uses a different approach to model the cross-task correlations.

The main contribution of this work is providing a framework for robotic learning tasks to leverage previously learned GP models, which can be especially valuable when limited training data are available for the new task. Reusing previously learned models allows past experience, while staying intact, to factor into new decisions, as and when relevant. This feedforward structure exploits the synergy between old and new learning tasks in keeping with the idea of lifelong learning. At the same time, the FAT-GP algorithm seamlessly handles situations where the environment undergoes a drastic change, and relearning from scratch is inevitable. Such a transfer learning algorithm would be useful while modeling time-varying fields like temperature, winds, and other dynamic phenomena.

II. FAT-GP: Forward Adaptive Transfer Using Gaussian Processes

A. Target Task Prediction

Consider two regression tasks S and T, which operate on input and response variables $\boldsymbol{x}$ and y of the same dimensionality and may be related. Additionally, we assume that S has S already been completed and a Gaussian process model $\mathcal{GP}_S$ with hyperparameters θ_S has been learned (see Appendix A). The goal of FAT-GP is to transfer relevant knowledge from source task to incomplete, target task T.

Gaussian processes make the smoothness assumption, whereby the response variables y_i have a Gaussian joint distribution. In keeping with this perspective, FAT-GP assumes that the source and target task labels are jointly distributed as

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

$$\begin{bmatrix} \boldsymbol{y}_T \\ \boldsymbol{y}_S \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}_T \\ \boldsymbol{m}_S \end{bmatrix}, \begin{bmatrix} C_{TT} & K_{TS} \\ K_{ST} & C_{SS} \end{bmatrix} \right) \tag{1}$$

where the mean and covariance matrices are denoted by $\boldsymbol{m}_{\text{joint}}$ and C_{joint}. Subscripts S and T represent components that are specific to source and target tasks, respectively; and superscript T is the transpose of the matrix. The ondiagonal block matrices are given by $C_{(\cdot)} = K_{(\cdot)} + \sigma^2_{n(\cdot)} I$, where the covariance $K_{(\cdot)}$ is defined by the choice of kernel function discussed later. C_{TT} and C_{SS} are square matrices with dimensions N_T and N_S, respectively, and are defined over the task-specific training samples X_S and X_T. $K_{ST} = K^T_{TS}$ are the cross-covariance matrices, where K_{TS} is a $N_T \times N_S$ rectangular matrix. Thus, matrix C_{joint} is a symmetric, square matrix with dimensionality $N_T + N_S$. Similarly, $\boldsymbol{m}_{\text{joint}}$ is a column vector of length $N_T + N_S$.

Because C_{joint} is a kernel covariance matrix, it must be positive semidefinite (PSD). One way to ensure that C_{joint} is PSD is to ignore task separation between the two datasets and define C_{TT}, C_{SS}, and K_{TS} using the same kernel function and hyperparameters. Let this formulation of C_{joint} be denoted by C_D, such that

$$C_D(x_i, x_j) = k(x_i, x_j; \theta) + \sigma_n^2 \delta_{ij} \quad \text{for } i, j = 1, \ldots, N_D \tag{2}$$

where k is any valid kernel function with hyperparameters θ; $N_D = N_T + N_S$; and x_i and x_j belong to $X_D = [X_T^T, X_S^T]^T$. This is equivalent to learning a single standard GP [1] by combining both tasks' datasets. However, this formulation gets rid of valuable context that the data come from two different tasks with different distributions.

FAT-GP maintains task boundaries by taking a modular approach with the design of the block diagonal matrices of the covariance matrix. FAT-GP capitalizes on this contextual information so that prediction for a certain task is based on all training data for that task, and the influence of data from the other potentially correlated task is controlled by a scaling factor $\lambda \in [-1, +1]$, which is one of the learned hyperparameters. Thus, the joint covariance matrix is formulated as

$$C_\tau(x_i, x_j) = \begin{cases} k_{\tau(T)}(x_i, x_j; \theta_{\tau(T)}) + \sigma^2_{n\tau(T)} \delta_{ij} & \text{when } x_i, x_j \in T \quad \text{and} \quad i, j = 1, \ldots, N_T \\ k_S(x_i, x_j; \theta_S) + \sigma^2_{nS} \delta_{ij} & \text{when } x_i, x_j \in S \quad \text{and} \quad i, j = 1, \ldots, N_S \\ \lambda k_{\text{cross}}(x_i, x_j; \theta_S, \theta_{\tau(T)}) & \text{when } x_i \in T \quad \text{and} \quad x_j \in S, \text{i.e., different tasks} \end{cases} \tag{3}$$

where $\theta_{\tau(T)}$ denotes target task hyperparameters learned during the FAT-GP training.

Because the kernel functions corresponding to the block diagonal matrices have different hyperparameters, the selection of the kernel function for the cross-covariance block matrices has to be made carefully to ensure that C_τ is positive semidefinite. Results presented in [38] define the kernel function for cross-covariance matrices as the convolution of the kernels used in the block diagonal matrices

$$k_{\text{cross}} = k_S * k_{\tau(T)} \tag{4}$$

We multiply the cross-covariance matrices by a scalar $\lambda \in [-1, +1]$, which is a measure of the similarity of the two tasks to give

$$K_{TS} = \lambda K_{\text{cross}} = K^T_{ST} \tag{5}$$

where the K_{cross} is the cross-covariance matrix with elements $K^{i,j}_{\text{cross}} = k_{\text{cross}}(x_i, x_j; \theta_S, \theta_{\tau(T)})$. Making use of theorem 1 from [33], it can be shown that the C_τ is PSD when the cross-covariance matrices K_{TS} and K_{ST} are defined as in Eq. (5).

The similarity measure, $|\lambda| \leq 1$, is an additional hyperparameter, which captures the correlation between the source and target task, and it is learned along with $\theta_{\tau(T)}$. When the source GP is a previously learned model, a value of λ close to zero signifies that past experience is obsolete, and the new model must be learned from scratch, possibly after extensive data collection. Thus, the joint covariance matrix for FAT-GP is given by

$$C_\tau = \begin{bmatrix} C_{TT} & K_{TS} \\ K_{ST} & C_{SS} \end{bmatrix} = \begin{bmatrix} C_{\tau(T)}(X_T, X_T) & \lambda K_{\text{cross},\tau(T)}(X_T, X_S) \\ \lambda K^T_{\text{cross},\tau(T)}(X_T, X_S) & C_S(X_S, X_S) \end{bmatrix} \tag{6}$$

where all the block matrices are functions of (source-specific-, task-specific-, and task-similarity-based) hyperparameters. Of these, the source hyperparameters are known a priori. The remaining unknown hyperparameters, denoted by $\theta_\tau = \{\theta_{\tau(T)}, \lambda\}$, are learned during training.

Consider, for example, a FAT-GP where both tasks use a Gaussian kernel with $\theta_S = \{\sigma^2_{fS}, \sigma^2_{nS}, L_S\}$ and $\theta_{\tau(T)} = \{\sigma^2_{fT}, \sigma^2_{nT}, L_T\}$, respectively. In both tasks, $\sigma^2_{f(\cdot)}$ represents the signal variance, whereas $L_{(\cdot)}$ is the diagonal matrix of length scales. Thus, the kernel functions for the ondiagonal matrices is given by

$$k_{(\cdot)}(x_i, x_j) = \sigma^2_{f(\cdot)} \exp\left[-\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{x}_j)^T L^{-1}_{(\cdot)} (\boldsymbol{x}_i - \boldsymbol{x}_j) \right] \tag{7}$$

and the cross-covariance kernel is given by

$$k_{\text{cross}}(\boldsymbol{x}_{T,i}, \boldsymbol{x}_{S,j}) = 2^{D/2} \sqrt{\sigma^2_{f\tau(T)} \sigma^2_{fS}} \frac{|L_{\tau(T)}|^{1/4} |L_S|^{1/4}}{|L_{\tau(T)} + L_S|^{1/2}} \exp\left[-\frac{1}{2} (\boldsymbol{x}_{T,i} - \boldsymbol{x}_{S,j})^T \left(\frac{L_{\tau(T)} + L_S}{2} \right)^{-1} (\boldsymbol{x}_{T,i} - \boldsymbol{x}_{S,j}) \right] \tag{8}$$

where D is the number of anisotropic input dimensions. If, however, the GP is isotropic (i.e., the same length scale is used for all input dimensions), the cross-covariance kernel would be given by

$$k_{\text{cross}}(\boldsymbol{x}_{T,i}, \boldsymbol{x}_{S,j}) = \sqrt{\sigma^2_{f\tau(T)} \sigma^2_{fS}} \sqrt{\frac{2 l_{\tau(T)} l_S}{l^2_{\tau(T)} + l^2_S}} \exp\left[-\frac{\|\boldsymbol{x}_{T,i} - \boldsymbol{x}_{S,j}\|^2}{l^2_{\tau(T)} + l^2_S} \right] \tag{9}$$

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

An important and notable characteristic of the cross-covariance kernel function in Eqs. (8) and (9) is that they are nonstationary [5,39] in the task domain, i.e., kernel function averages over the length scales local to each of the tasks. As a result, this kernel function implicitly captures the covariance between two $\boldsymbol{x}$ inputs as a combination of the characteristics of both tasks.

Once the FAT-GP is trained and the hyperparameters are known (as explained in Sec. III.B), it can be used to predict the target task response y'_T for any unseen input variable $\boldsymbol{x}'$. The joint distribution of y' with $\boldsymbol{y}_T$ and $\boldsymbol{y}_S$ can be written by calculating the $\boldsymbol{x}'$ correlation with the target as well as source training samples:

$$\begin{bmatrix} y' \\ \boldsymbol{y}_T \\ \boldsymbol{y}_S \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_{\tau(T)}(\boldsymbol{x}') \\ \boldsymbol{m}_{\tau(T)}(X_T) \\ \boldsymbol{m}_S(X_S) \end{bmatrix}, \begin{bmatrix} k_\tau(\boldsymbol{x}', \boldsymbol{x}') & k_\tau(\boldsymbol{x}', X_T) & k_\tau(\boldsymbol{x}', X_S) \\ k_\tau(\boldsymbol{x}', X_T)^T & C_{TT} & K_{TS} \\ k_\tau(\boldsymbol{x}', X_S)^T & K_{ST} & C_{SS} \end{bmatrix} \right) \tag{10}$$

The kernel functions k_τ are interpreted based on the task assignment of their two inputs. Hence, because $\boldsymbol{x}'$ belongs to the target task, based on Eq. (3),

$$\begin{aligned} k_\tau(\boldsymbol{x}', \boldsymbol{x}') &= k_{\tau(T)}(\boldsymbol{x}', \boldsymbol{x}') \\ k_\tau(\boldsymbol{x}', X_T) &= k_{\tau(T)}(\boldsymbol{x}', X_T) \\ k_\tau(\boldsymbol{x}', X_S) &= \lambda k_{\text{cross}}(\boldsymbol{x}', X_S) \end{aligned} \tag{11}$$

Consequently, the prediction for y' is given by its conditional distribution

$$p(y'|\boldsymbol{x}', \boldsymbol{y}_T, X_T, \boldsymbol{y}_S, X_S, \theta_\tau, \theta_S) = \mathcal{N}(\mu'_\tau, C'_\tau)$$

where

$$\begin{aligned} \mu'_\tau &= \boldsymbol{m}_{\tau(T)}(\boldsymbol{x}') + [\, k_\tau(\boldsymbol{x}', X_T) \quad k_\tau(\boldsymbol{x}', X_S)\,] C_\tau^{-1} \left(\begin{bmatrix} \boldsymbol{y}_T \\ \boldsymbol{y}_S \end{bmatrix} - \begin{bmatrix} \boldsymbol{m}_{\tau(T)}(X_T) \\ \boldsymbol{m}_S(X_S) \end{bmatrix} \right) \\ C'_\tau &= k_\tau(\boldsymbol{x}', \boldsymbol{x}') + \sigma^2_{n\tau(T)} - [\, k_\tau(\boldsymbol{x}', X_T) \quad k_\tau(\boldsymbol{x}', X_S)\,] C_\tau^{-1} \begin{bmatrix} k_\tau(\boldsymbol{x}', X_T) \\ k_\tau(\boldsymbol{x}', X_S) \end{bmatrix} \end{aligned} \tag{12}$$

Equations (10–12) easily scale from predictions for individual target inputs $\boldsymbol{x}'$ to individual or joint prediction for multiple target inputs X'.

B. Hyperparameter Learning

The unknown hyperparameters in Eqs. (5) and (12) are $\theta_\tau = \{\theta_{\tau(T)}, \lambda\}$, where τ represents a forward transfer inference, and $\theta_{\tau(T)}$ denotes target task hyperparameters learned by FAT-GP. Unlike the conventional GP, FAT-GP learns the hyperparameters by maximizing the log marginal likelihood of only the target response variables given the source data:

$$\theta^*_\tau = \arg\max_{\theta_\tau} \ln\, p(\boldsymbol{y}_T|\boldsymbol{y}_S, X_T, X_S, \theta_S, \theta_\tau) \tag{13}$$

Using Bayes's rule on Eq. (1), the marginal distribution of the response variables of the target task $\boldsymbol{y}_T$ is conditionally inferred from the source task as follows:

$$p(\boldsymbol{y}_T|\boldsymbol{y}_S, X_T, X_S, \theta_T, \theta_S) = \mathcal{N}(\mu_{T|S}, C_{T|S}) \tag{14}$$

where

$$\begin{aligned} \mu_{T|S} &= \boldsymbol{m}_T + K_{TS} C_{SS}^{-1} (\boldsymbol{y}_S - \boldsymbol{m}_S) \\ C_{T|S} &= C_{TT} - K_{TS} C_{SS}^{-1} K_{ST} \end{aligned}$$

This approach mitigates two problems encountered when maximizing the log likelihood of the joint distribution of the source and target [Eq. (1)]. First, it avoids calculation and inversion of an $N_D = N_S + N_T$ size covariance matrix at each iteration of the maximization, instead calculating and inverting $C_{T|S}$, for which $N_T \times N_T$. Second, and more importantly, because the source has already been learned, it focuses the learning on the target.

Using the multivariate normal distribution from Eq. (14) in Eq. (13), the maximum likelihood estimator (MLE) is written as

$$\begin{aligned} \theta^*_\tau &= \arg\max_{\theta_\tau} \ell_n \left[\frac{1}{2\pi^{N_T/2} |C_{T|S}|^{1/2}} \exp\left\{ -\frac{1}{2} (\boldsymbol{y}_T - \mu_{T|S})^T C_{T|S}^{-1} (\boldsymbol{y}_T - \mu_{T|S}) \right\} \right] \\ &= \arg\max_{\theta_\tau} \left[-\frac{1}{2} (\boldsymbol{y}_T - \mu_{T|S})^T C_{T|S}^{-1} (\boldsymbol{y}_T - \mu_{T|S}) - \frac{1}{2} \ell_n\, |C_{T|S}| - \frac{N_T}{2} \ell_n\, 2\pi \right] \end{aligned} \tag{15}$$

The first term in Eq. (15) is the Mahalanobis distance between the observed target response variables $\boldsymbol{y}_T$ and the FAT-GP predictive distribution. It quantifies the empirical risk of the learned FAT-GP. The second term is a regularization term that prevents overfitting. The maximum likelihood estimator finds θ^*_τ by trading off between these two components.

MLE can be performed using a gradient descent optimizer. This requires the computation of the derivative of the log marginal likelihood with respect to each of the hyperparameters in θ_τ [1,40]. Note that the covariance matrix is dependent on the hyperparameters, but the response variables $\boldsymbol{y}_T$ are not:

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

Table 1 Computational cost comparison between FAT-GP (where $N_S > N_T$) and symmetric multitask learning (e.g., AT-GP)

Source task learning	Per iteration cost of computing C_{SS}^{-1}	Training cost
Simultaneously with target task [33]	$O(N_S^3)$	$O(N_S^3 + N_T^3)$
Completed previously (FAT-GP)	Precomputed	$O(N_T^3 + N_S N_T)$

$$\frac{\partial}{\partial \theta_\tau} \ell_n\, p(\boldsymbol{y}_T|\theta_\tau) = -2\left[\frac{1}{2}(\boldsymbol{y}_T - \mu_{T|S})^T C_{T|S}^{-1} \frac{\partial(\boldsymbol{y}_T - \mu_{T|S})}{\partial \theta_\tau}\right] - \frac{1}{2}(\boldsymbol{y}_T - \mu_{T|S})^T \frac{\partial C_{T|S}^{-1}}{\partial \theta_\tau}(\boldsymbol{y}_T - \mu_{T|S}) - \frac{1}{2}\frac{\partial\, \ell_n\, |C_{T|S}|}{\partial \theta_\tau}$$

which can be reduced to

$$\frac{\partial}{\partial \theta_\tau} \ell_n\, p(\boldsymbol{y}_T|\theta_\tau) = (\boldsymbol{y}_T - \mu_{T|S})^T C_{T|S}^{-1} \frac{\partial \mu_{T|S}}{\partial \theta_\tau} + \frac{1}{2}(\boldsymbol{y}_T - \mu_{T|S})^T C_{T|S}^{-1} \frac{\partial C_{T|S}}{\partial \theta_\tau} C_{T|S}^{-1}(\boldsymbol{y}_T - \mu_{T|S}) - \frac{1}{2} Tr\left(C_{T|S}^{-1} \frac{\partial C_{T|S}}{\partial \theta_\tau}\right) \tag{16}$$

The cost for each iteration of the training optimization comprises calculating $C_{T|S}$ and then inverting it. Equation (14) shows that the calculation of $C_{T|S}$ is dominated by the inversion of C_{SS}. If both θ_S and θ_T are unknown, the cost of evaluating Eq. (15) is $O(N_S^3 + N_T^3)$, which is dominated by the $O(N_S^3)$ cost of inverting the square matrix C_{SS} of dimension $N_S (> N_T)$. However, in the FAT-GP formulation, θ_S is known a priori. Hence, C_{SS}^{-1} can be precomputed. This reduces the computational complexity to $O(N_T^3 + N_S N_T)$. The final cost savings will depend on the relative sizes of the source and task training sets. Thus, rehashing transfer learning in the context of robot-based lifelong learning changes the computational requirements, as summarized in Table 1.

C. Negative Transfer

Negative transfer is the phenomenon where providing other tasks to help learning actually hurts the learning of the target task [37]. Negative transfer occurs when the transfer learning approach assumes a relationship between tasks that is not correct. As a result, the transfer learning process correlates nonrelated characteristics of the source task to the target task. The FAT-GP algorithm is not guaranteed to prevent negative transfer, but it includes mechanisms that can mitigate it.

FAT-GP mitigates negative transfer through two mechanisms, but it still enables it by a third. First, FAT-GP assumes the target task has different hyperparameters than the source tasks. These hyperparameters are learned during the transfer process. It is possible that the target task learns the same hyperparameters as the source, but it is not required. Thus, the resulting hyperparameters of the target task are not constrained by the source. Second, the similarity measure λ describes the correlation between tasks. A value of $\lambda = 0$ means the source task data have no correlation with the target task and would not be used in predicting target task outputs. Because λ is learned as part of the transfer, FAT-GP always has the option of ignoring the source data if a single task GP describes the target training data best. Given these two mitigations, the results of the transfer learning are still influenced by the assumed structure of the cross-task covariance K_{TS} and the cross-task kernel function k_{cross}. For $\lambda \neq 0$, the kernel function and associated hyperparameters of the source tasks are coupled to the target task kernel. As a result, the learned hyperparameters will be influenced by the source task. Overall, empirical results have yielded minimal to no negative transfer. The assessment of negative transfer is included in the simulation results provided in Sec. V.

It should be noted that the FAT-GP algorithm is also heavily dependent on the training data collected from the target task. Negative transfer could occur if the relatively small dataset for the target task (compared to the data collected from the source task) is not representative of the relationship between tasks. However, this issue is an instance of the more fundamental problem of learning a Gaussian process over sufficient data and is not limited to FAT-GP.

III. Simulation Examples

A. FAT-GP Demonstration via One-Dimensional Problems

This section uses one-dimensional (1-D) problems to illustrate how FAT-GP can harness previously learned models for learning new models efficiently. More important, it highlights the improvement in performance that FAT-GP provides over the target GP, i.e., GP learned with the limited target training samples. By selecting two 1-D signals that are related through transformation and where the similarity can be visually verified, it is clear how forward adaptive transfer for Gaussian process regression reduces the target task's validation error in spite of the limited amount of training data.

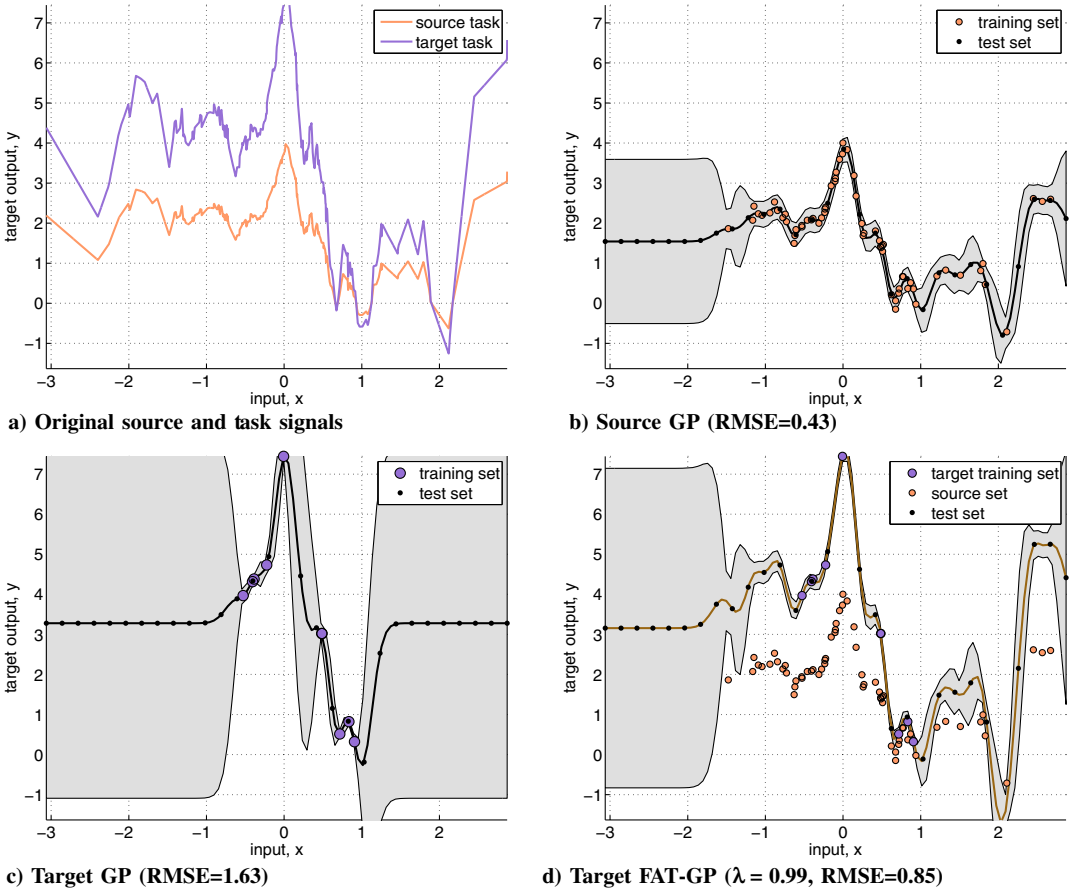
In this example, shown in Fig. 2, the target and source tasks are related through an affine transformation. Figure 2a shows that the target signal y_T is obtained by scaling the source task y_S by a factor of two. The source signal is learned using a Gaussian process regression on a training set of 65 samples. The output is $\mathcal{GP}_S$, which is a nonparametric model for the source task. The mean and variance of the model are represented by the solid black line and the shaded gray area in Fig. 2b, whereas the training samples are denoted by light dots. Finally, the points in the validation set are denoted by smaller dark dots. Testing the prediction performance of $\mathcal{GP}_S$ on this validation set of 30 samples results in a root-mean-squared error (RMSE) of 0.43.

The target learning task, on the other hand, has access to only 10 training samples, resulting in a high validation RMSE of 1.63. As shown in Fig. 2c, the $\mathcal{GP}_T$ learns the peak and the trough near $x = 0$ and $x = 1$, respectively. However, due to the limited number of training samples, it fails to capture any of the features for $x \leq -1$ and $x \geq 1$. In fact, the high variance at these x values signifies the GP's lack of confidence in these predictions. Combining these 10 target samples with the 65 source samples and results in the FAT-GP shown in Fig. 2d, which has a much lower RMSE of 0.85.

The FAT-GP algorithm learns the hyperparameters by combining the 10 target training samples, 65 source training samples, and source model $\mathcal{GP}_S$. During this training process, it learns that the similarity between tasks T and S is $\lambda = 0.99$. Consequently, the source task heavily contributes to target task, and the resulting FAT-GP is shown in Fig. 2d. This FAT-GP's prediction achieves a much higher fidelity with the original target signal in Fig. 2a, and this is reflected in the low validation RMSE of 0.85. In addition, the FAT-GP is also more confident in its estimates having based its inference on a larger dataset of 75 samples.

As this example shows, including data from a similar task helps the GP regression confidently predict for x values not captured in its own dataset. On one hand, this makes up for the low density of training data in regions of the target task that have been poorly sampled (for example, between -1 and 1 on the X axis), and it reduces the uncertainty in the prediction. On the other hand, perhaps more importantly, the transfer provides the target task information about completely unexplored regions of its task space, such as the regions between one and two. As expected, the improvement does not extend to the regions where neither task has sampled. This can be seen by observing that FAT-GP continues to have a high prediction variance between and -2, even after the forward adaptive transfer.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437



a) Original source and task signals

b) Source GP (RMSE=0.43)

c) Target GP (RMSE=1.63)

d) Target FAT-GP ($\lambda = 0.99$, RMSE=0.85)

Fig. 2 Example where the target task T is a scaling of source task S, i.e., $y_T = 2y_S$.

B. FAT-GP for Two-Dimensional Models

This section uses a simulation study to illustrate how FAT-GP would be useful in learning two-dimensional models in the presence of little training data and limited exploration. In particular, this study is motivated by the problem of learning communication models for unmanned aircraft [11,12,17,24]. Such models are useful for a variety of applications such as spectrum characterization, radio emitter detection and tracking, airborne meshed network formation, and communication-aware information gathering.

The source and target task are setup such that

$$\boldsymbol{y}_S = f(X_S) + v_S \tag{17}$$

$$\boldsymbol{y}_T = g(f(X_T)) + v_T \tag{18}$$

where the first term represents the mean field, and the second term represents the geospatial variations. The variations v_S and v_T are sampled from the same distribution $\mathcal{N}(0, K_{\text{variation}})$. Because we are motivated by communication modeling, the mean field $f(X_S)$ describes an exponential path loss and the variations represent interference or fading [27]. In Fig. 3, which shows the source and target tasks, their mean fields, and variations, $g(\cdot) = 2 \times (\cdot)$. The source GP is learned using a dense set of 2500 training samples, as shown in Fig. 4a. The mean prediction of this source GP achieves a high fidelity with the original source task, which can be seen by comparing Fig. 4b with Fig. 3a.

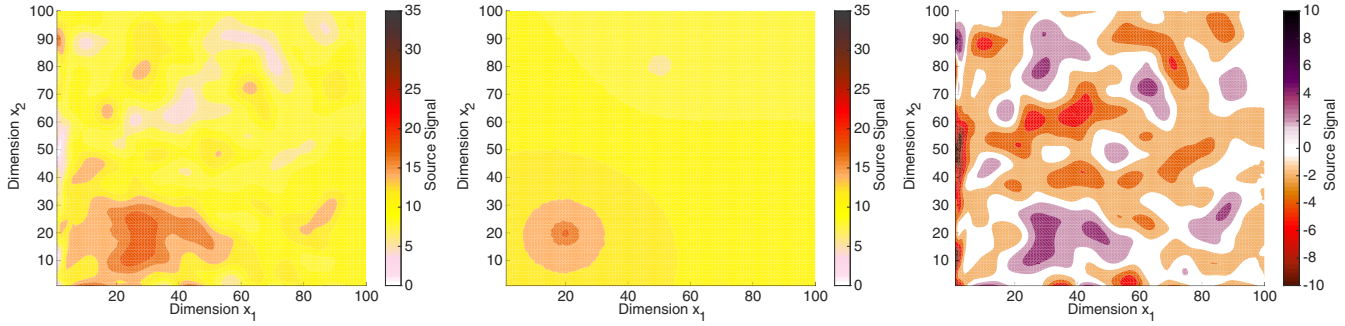
Figure 5 presents two scenarios for learning the target task with a limited training set of size 50. In Fig. 5a, the samples are spread over the entire environment, representing a situation where a low density of data is available. Figure 5b, on the other hand, represents the situation where, due to the large scale of environments, target data are only available for part of the environment. In both these cases, FAT-GP can leverage the high-fidelity source GP, which has captured inherent characteristics of the environment, and learn better models until more target data can be collected.

Figures 6 and 7 show a comparison of the target GPs and FAT-GPs learned using the two datasets shown in Fig. 5. In the first example, where samples are available throughout the entire environment, the target GP shows an RMSE of 3.58. In spite of the high error, its confidence is high in most of the region. Due to similarity between the tasks, FAT-GP learns a value of $\lambda = 0.94$ during the training. Using the source GP and source data in its predictions, the FAT-GP reduces the RMSE to 2.97. Seeing the variability of the larger source dataset also causes the FAT-GP to adapt its uncertainty so that it has very high confidence only locally around the target samples. The resulting predictive distribution outperforms the overconfident target GP. Consequently, the mean standardized log loss (MSLL) of the FAT-GP with the target GP as the baseline is -0.2276.

In the second example, because the environment is observed only partially, the RMSE and variance predictions are higher than in the first example. The higher RMSE of 5.75 is mostly due to the right half of the environment, where the target GP predicts a flat field with a value equal to the mean of the observed y_T. As this region is outside the distance of correlation for all the training data, it also has the highest variance. The FAT-GP, which learns $\lambda = 0.96$ and transfers source data proportionally, reduces the uncertainty of the unobserved half of the region, and it achieves an RMSE of 4.46. Once again, because it outperforms the overconfidence target GP in terms of accuracy and uncertainty, the MSLL of the FAT-GP with respect to the target GP is -0.11.

In both of these cases, the FAT-GP makes use of the correlation between the source and target tasks to make up for the limited information in the small target dataset. In this manner, previous observations and models of the environment can help inform model updates, especially as the

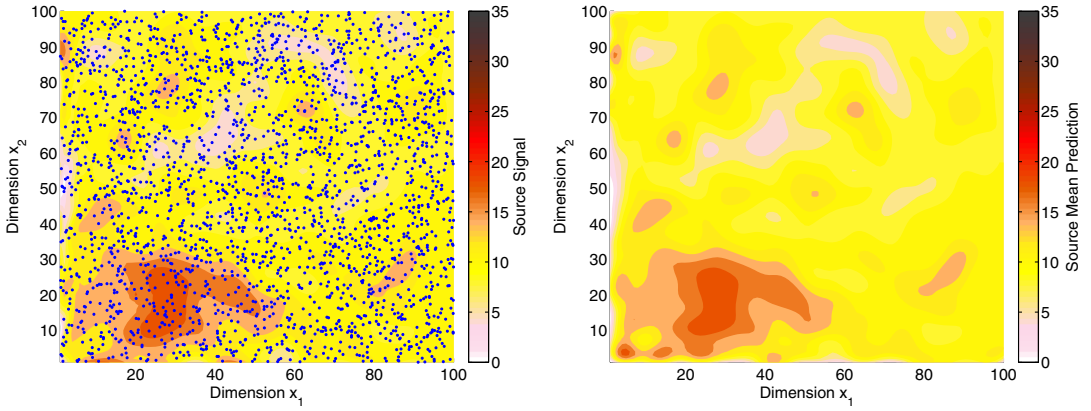
Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437



a) Source b) Source mean field c) Source noise

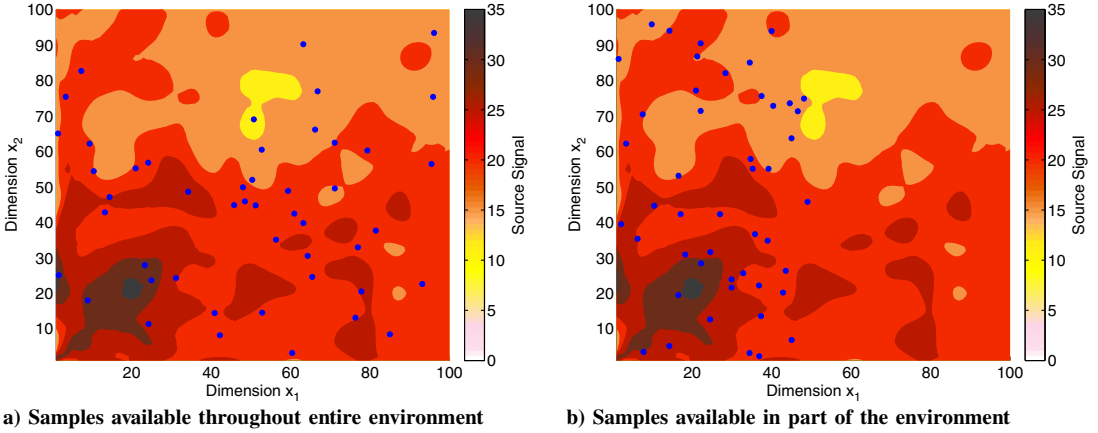
d) Target e) Target mean field f) Target noise

Fig. 3 True environments in source and target tasks.



a) Source samples b) Source GP mean predictions

Fig. 4 Source GP learned using 2500 samples of the source task.

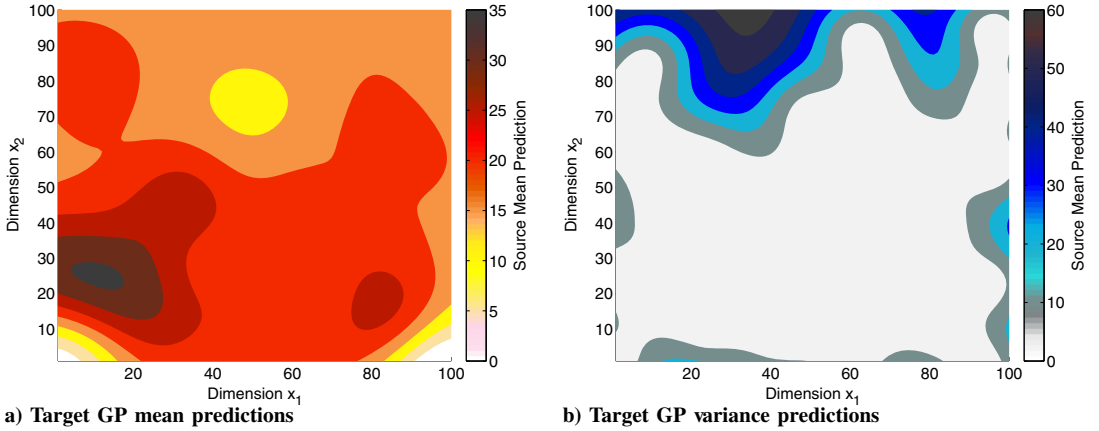


a) Samples available throughout entire environment b) Samples available in part of the environment

Fig. 5 Two scenarios with limited target training set of size 50 available.

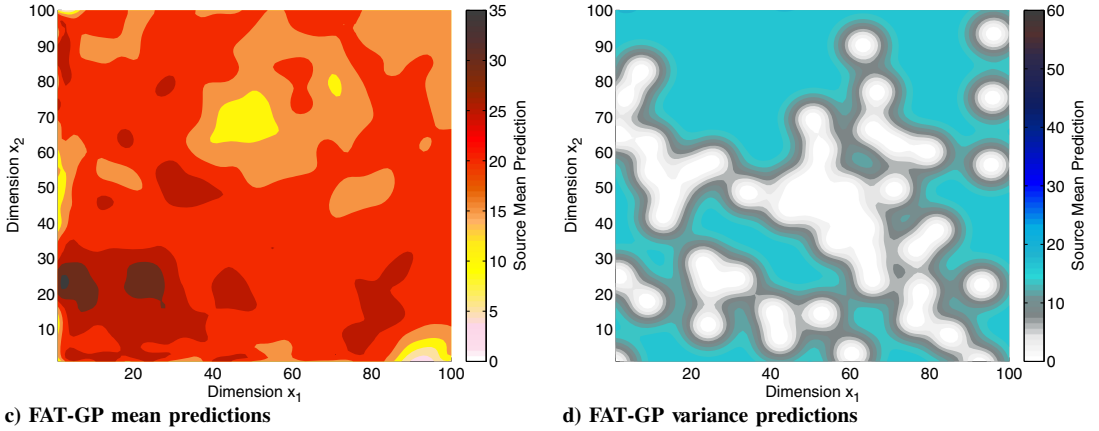
Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437



a) Target GP mean predictions

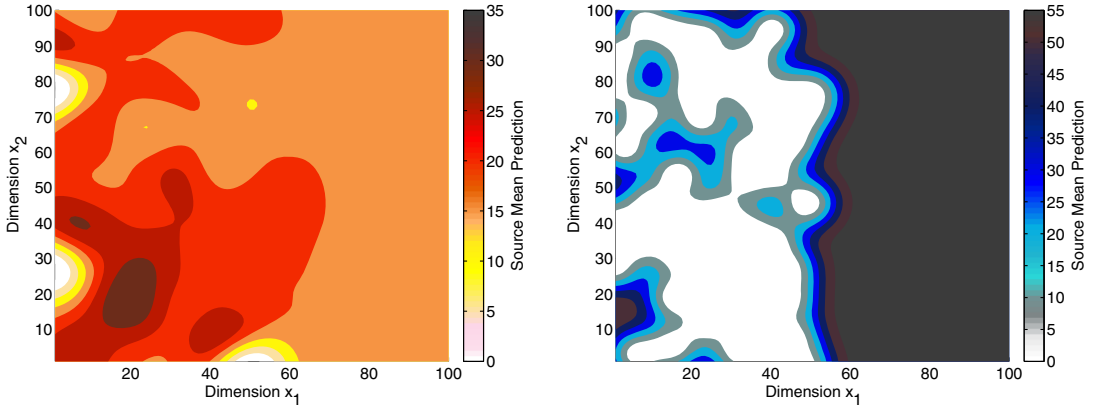
b) Target GP variance predictions



c) FAT-GP mean predictions

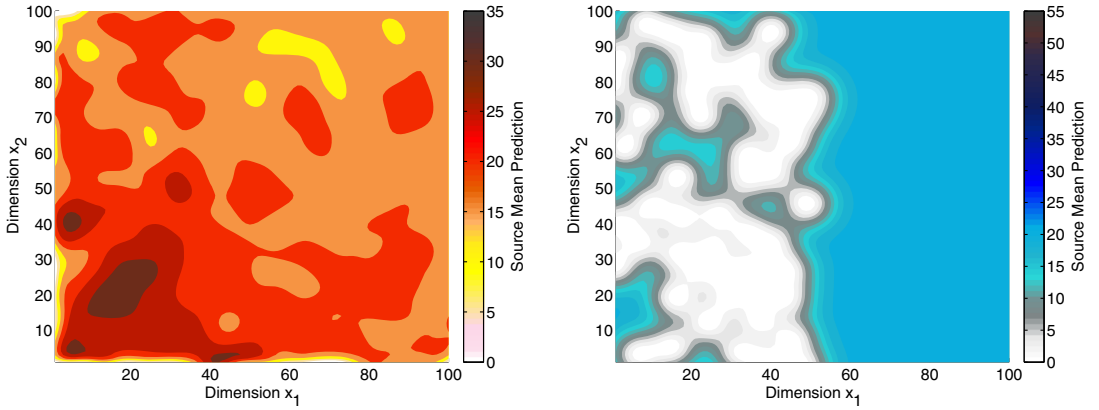
d) FAT-GP variance predictions

Fig. 6 Target GP and FAT-GP learned using 50 samples of the target task taken over the entire environment.



a) Target GP mean predictions

b) Target GP variance predictions



c) FAT-GP mean predictions

d) FAT-GP variance predictions

Fig. 7 Target GP and FAT-GP learned using 50 samples of the target task taken over half the environment.

environment is being explored for updated observations. However, efficient and beneficial transfer largely depends on the locations at which target samples are taken. The choice of target locations not only affects the information they provide about the task itself but also helps learn the correlations between the tasks effectively. Hence, future work must focus on transfer planning, which is active learning based on the combined objectives of transfer and target learning.

IV. Relationship Between FAT-GP and Target GP

The FAT-GP combines information from the target with transferred information from the source to learn a model of the target task. To understand what is transferred, how the target and transfer components interact, and when the transfer boosts the target performance, it is important to understand the relationship between FAT-GP with the conventional target GP.

The target GP learns the target task using only the limited target task training data, contains the standard GP hyperparameters, and is trained using maximum likelihood estimation on the target data:

Hyperparameters:

$$\theta_T = \{\sigma_{fT}^2, \sigma_{nT}^2, L_T, \boldsymbol{m}_T\} \tag{19}$$

Training:

$$\theta_T^* = \arg\max_{\theta_T} \ln\, p(\boldsymbol{y}_T | X_T, \theta_T)$$

Similarly, the prediction for target test sample $\boldsymbol{x}'$ is given by the standard GP equations:

$$\mu_T(\boldsymbol{x}') = \boldsymbol{m}_T(\boldsymbol{x}') + k_T(\boldsymbol{x}', X_T) C_T(X_T, X_T)^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T))$$
$$\sigma_T^2(\boldsymbol{x}') = k_T(\boldsymbol{x}', \boldsymbol{x}') + \sigma_{nT}^2 - k_T(\boldsymbol{x}', X_T) C_T(X_T, X_T)^{-1} k_T(X_T, \boldsymbol{x}')$$

This GP provides a baseline for the performance that can be obtained without transfer. On the other hand, FAT-GP τ uses a priori source hyperparameters, and then it learns $\theta_{\tau(T)}$ and λ using source and target data.

A. Impact of Transfer on Mean and Variance Prediction

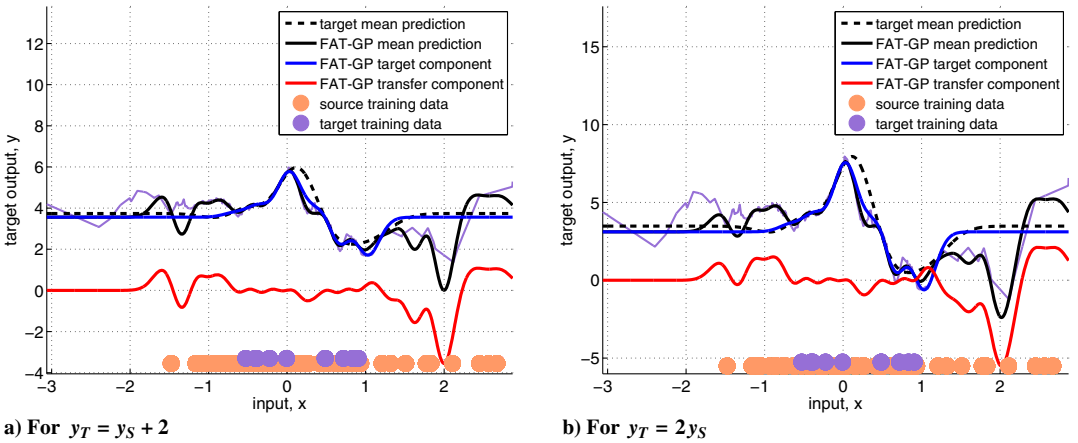
To investigate the benefits of transfer, we analyze how the mean predictions from both these GPs compare. Using properties of positive semidefinite matrices and equations defined in Sec. III, it can be shown that the FAT-GP mean prediction comprises two components:

$$\mu_\tau' = \underbrace{\begin{pmatrix} \boldsymbol{m}_\tau(\boldsymbol{x}') + \\ k_\tau(\boldsymbol{x}', X_T) C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_\tau(X_T)) \end{pmatrix}}_{\text{Target component}} + \underbrace{\begin{matrix} [k_\tau(\boldsymbol{x}', X_T) C_{TT}^{-1} K_{TS} - k_\tau(\boldsymbol{x}', X_S)] \times \\ [C_{SS} - K_{ST} C_{TT}^{-1} K_{TS}]^{-1} \times \\ [K_{ST} C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_\tau(X_T)) - (\boldsymbol{y}_S - \boldsymbol{m}_S(X_S))] \end{matrix}}_{\text{Transfer component}} \tag{20}$$

Figure 8 shows these target and transfer components as solid lines. The mean predictions for FAT-GP and target GP are also shown using solid and dashed lines. The target component is very close to the dashed target GP prediction because they both represent the information that is available in the target training set, shown by darker dots at the bottom of the figure. The transfer component is primarily responsible for the differences between the FAT-GP and the target GP. Note that its major contributions are in regions where no target samples are available; the gaps are filled by transferring from the source GP. Finally, for $x \leq -2$, where neither source nor target data are available, the FAT-GP cannot provide meaningful predictions.

Analogous to Eq. (20), the variance prediction of the FAT-GP also comprises two components:

$$\sigma_\tau^2(\boldsymbol{x}') = \underbrace{\begin{pmatrix} k_\tau(\boldsymbol{x}', \boldsymbol{x}') + \sigma_{n\tau(T)}^2 - \\ k_\tau(\boldsymbol{x}', X_T) C_{TT}^{-1} k_\tau(X_T, \boldsymbol{x}') \end{pmatrix}}_{\text{Target Component}} - \underbrace{\begin{matrix} [k_\tau(\boldsymbol{x}', X_T) C_{TT}^{-1} K_{TS} - k_\tau(\boldsymbol{x}', X_S)] \times \\ [C_{SS} - K_{ST} C_{TT}^{-1} K_{TS}]^{-1} \times \\ [K_{ST} C_{TT}^{-1} k_\tau(X_T, \boldsymbol{x}') - k_\tau(X_S, \boldsymbol{x}')] \end{matrix}}_{\text{Transfer Component}} \tag{21}$$



a) For $y_T = y_S + 2$ **b) For $y_T = 2y_S$**

Fig. 8 FAT-GP mean prediction components and its comparison to target GP.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

which can also be viewed as

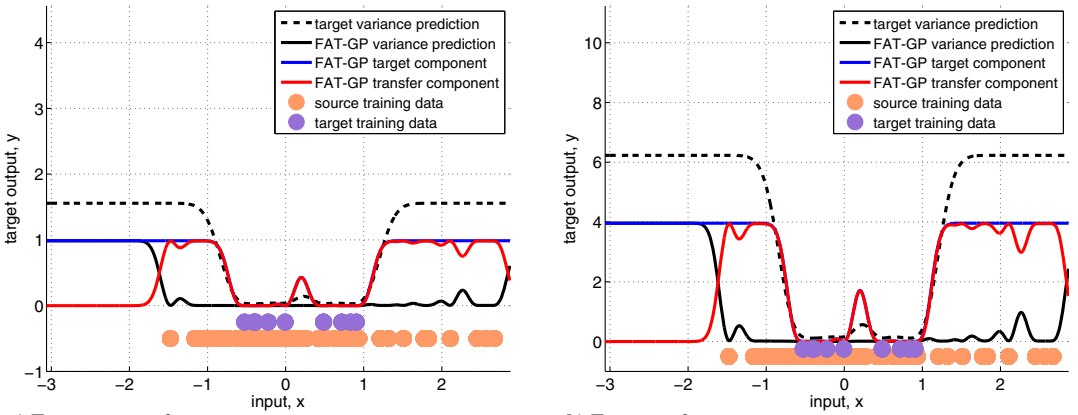
$$\sigma_\tau^2(\boldsymbol{x}') = \underbrace{\sigma_{f\tau(T)}^2 + \sigma_{n\tau(T)}^2}_{\text{prior variance}} - \underbrace{k_\tau(\boldsymbol{x}', X_T) C_{TT}^{-1} k_\tau(X_T, \boldsymbol{x}')}_{\text{Reduction in uncertainty due to target data}}$$
$$- \underbrace{[k_\tau(\boldsymbol{x}', X_T) C_{TT}^{-1} K_{TS} - k_\tau(\boldsymbol{x}', X_S)][C_{SS} - K_{ST} C_{TT}^{-1} K_{TS}]^{-1}[K_{ST} C_{TT}^{-1} k_\tau(X_T, \boldsymbol{x}') - k_\tau(X_S, \boldsymbol{x}')]}_{\text{Reduction in uncertainty due to transfer}} \tag{22}$$

The perspectives of Eqs. (21) and (22) are illustrated in Figs. 9 and 10, respectively. Note that, in Fig. 9, the FAT-GP variance prediction represented by the darkest solid line is obtained by subtracting FAT-GP transfer component from the FAT-GP target component. The target component of the variance prediction or uncertainty is low or zero where target samples are available, and it rises to $\sigma_{f\tau(T)}^2 + \sigma_{n\tau(T)}^2$ elsewhere. The transfer component represents a reduction in uncertainty due to transferring, and analogously provides zero correction where no source data are available.

More interestingly, however, it also provides zero correction where target samples are present. This alternating behavior is more clearly apparent in Fig. 10, in which the components from Eq. (22) are represented. Here, the FAT-GP target component and FAT-GP transfer component are both subtracted from the dashed line, reducing the uncertainty to give the solid FAT-GP variance prediction. Thus, the reduction in uncertainty from the transfer counters only the uncertainty not corrected by the target data itself.

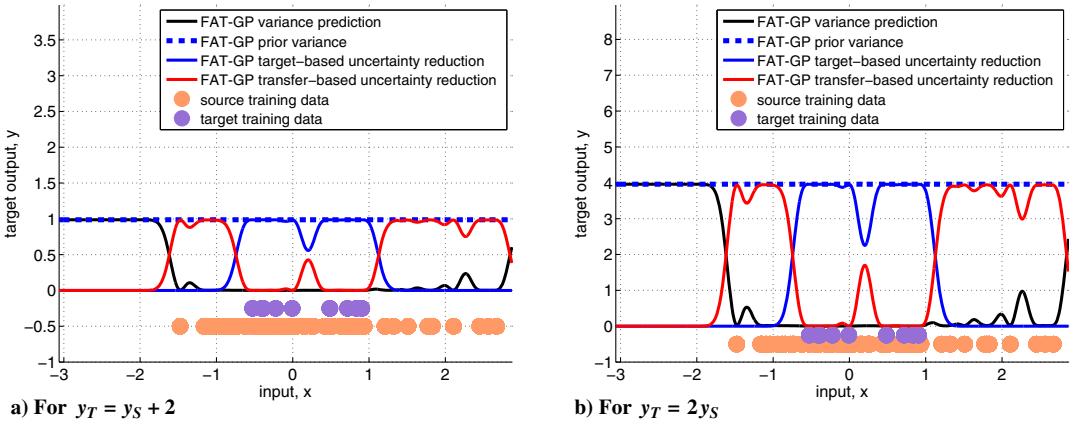
Analyzing how transfer behaves based on the relative positions of the target and source data samples reveals a link to the signal-to-noise ratio (SNR) of the target task. As shown in Appendix C, how a source sample $\boldsymbol{x}_S$ and a target sample $\boldsymbol{x}_T$ reduce the uncertainty at an unseen sample $\boldsymbol{x}'$ depends on their relative positions, and it is captured in the coefficient of transfer b given by

$$b = \begin{cases} \dfrac{\sigma_{n\tau(T)}^2}{\sigma_{n\tau(T)}^2 + \sigma_{f\tau(T)}^2} & \text{when source and target are colocated at } \boldsymbol{x}' \\ \dfrac{c_{TS} c_{\tau(T)} \sigma_{f\tau(T)}^2}{\sigma_{n\tau(T)}^2 + \sigma_{f\tau(T)}^2} & \text{when target is close to } \boldsymbol{x}' \\ -c_S & \text{when only a source is close to } \boldsymbol{x}' \end{cases} \tag{23}$$



a) For $y_T = y_S + 2$ **b) For $y_T = 2y_S$**

Fig. 9 FAT-GP variance prediction components and its comparison to target GP.



a) For $y_T = y_S + 2$ **b) For $y_T = 2y_S$**

Fig. 10 Reduction of uncertainty in FAT-GP.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

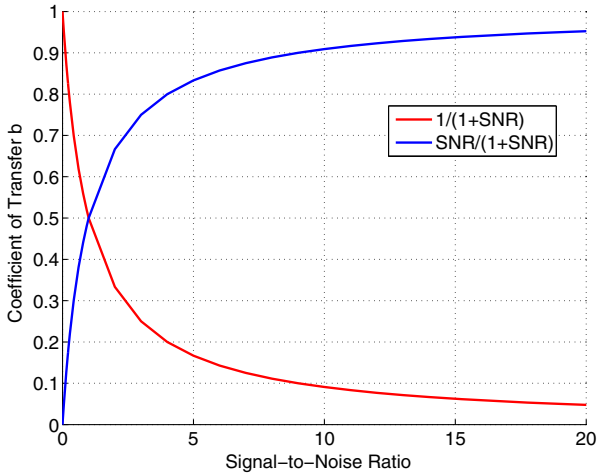


Fig. 11 Coefficient of transfer b as a function of the SNR.

where

$$c_{TS} = \exp\left[-\frac{\|\boldsymbol{x}_T - \boldsymbol{x}_S\|^2}{l_{\tau(T)}^2 + l_S^2}\right]$$

$$c_S = \exp\left[-\frac{\|\boldsymbol{x}' - \boldsymbol{x}_S\|^2}{l_{\tau(T)}^2 + l_S^2}\right]$$

$$c_{\tau(T)} = \exp\left[-\frac{\|\boldsymbol{x}_T - \boldsymbol{x}'\|^2}{2l_{\tau(T)}^2}\right]$$

Note that, here, "close to" implies being within the distance of correlation.

Because $\sigma_{f\tau(T)}^2$ and $\sigma_{n\tau(T)}^2$ represent the signal and noise variances, respectively, $\sigma_{f\tau(T)}^2/\sigma_{n\tau(T)}^2$ represents the signal-to-noise ratio of the target portion of the FAT-GP. Hence, the first two cases in Eq. (23) can also be viewed as

$$b = \begin{cases} \dfrac{1}{1+\text{SNR}} & \text{when source and target are colocated at } \boldsymbol{x}' \\ \dfrac{c_{TS}c_{\tau(T)}\text{SNR}}{1+\text{SNR}} & \text{when target is close to } \boldsymbol{x}' \end{cases} \tag{24}$$

Under this perspective, as shown in Fig. 11, when SNR < 1, the source will contribute heavily to predictions even when target data are available in the same region. On the other hand, when SNR > 1, for colocated source and target data points, the contribution is muted with a growing SNR. As the distance between the source data point and $\boldsymbol{x}'$ grows, the influence of the transfer is routed via the target data points close to $\boldsymbol{x}'$, and it is dictated by a combination of the SNR and the correlation between the tasks.

B. Dissimilar Source and Target Tasks

When the target and source are dissimilar (i.e., $\lambda = 0$), all cross-covariance terms go to zero. Then, the difference between the two GPs' mean predictions is given by

$$\mu_\tau' = \boldsymbol{m}_{\tau(T)}(\boldsymbol{x}') + k_{\tau(T)}(\boldsymbol{x}', X_T)C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_{\tau(T)}(X_T)) \tag{25}$$

Comparing this against Eq. (20), we see that Eq. (25) has the same form as the mean prediction for the target GP. These predictions would be equal if $\theta_{\tau(T)} = \theta_T$, i.e., target hyperparameters learned in both GPs are identical. To investigate this, we revisit the log marginal likelihood that the FAT-GP maximizes:

$$\theta_\tau^* = \arg\max_{\theta_T} -\frac{1}{2}(\boldsymbol{y}_T - \mu_{T|S})C_{T|S}^{-1}(\boldsymbol{y}_T - \mu_{T|S}) - \frac{1}{2}\ell_n|C_{T|S}| - \frac{N_T}{2}\ell_n 2\pi \tag{26}$$

When λ is set to zero, from Eq. (14), we see that $K_{TS} = \mathbf{0} = K_{ST}$, and this maximization becomes

$$\theta_\tau^* = \arg\max_{\theta_T} -\frac{1}{2}(\boldsymbol{y}_T - \boldsymbol{m}_T)^T C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T) - \frac{1}{2}\ell_n|C_{TT}| - \frac{N_T}{2}\ell_n 2\pi \tag{27}$$

which is the maximization used by the target GP to learn the hyperparameters. Therefore, the remaining hyperparameters $\theta_{\tau(T)}$ that are learned are the same as θ_T learned by the target GP. Hence, the difference between their mean predictions is given by

$$\mu_\tau' - \mu_T' = 0 \tag{28}$$

Thus, when the target and source task have no similarity (i.e., $\lambda = 0$), the learned FAT-GP is in fact the same as the target GP.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

V. Comparing the FAT-GP and Target GP

A. RMSE and MSLL Comparisons

Using an obsolete or incorrect model can have adverse effects on the performance of the mission. This section uses the root-mean-squared error and mean standardized log loss to quantify how different learning schemes involving source and target tasks compare to the FAT-GP. We enumerate these different learning configurations here:

1) The first configuration is the source GP S: Only the source data $\{X_S, \boldsymbol{y}_S\}$ are used for training and prediction. This configuration maps to the scenario where a previously learned GP model is considered plausible for the current mission and used as is. Comparing this configuration to FAT-GP will help illustrate the drawbacks of using outdated models:

Hyperparameters:

$$\theta_S = \{\sigma_{fS}^2, \sigma_{nS}^2, L_S, \boldsymbol{m}_S\}$$

Training:

$$\theta_S^* = \arg\max_{\theta_S} \ell_n \; p(\boldsymbol{y}_S | X_S, \theta_S)$$

Prediction:

$$\begin{aligned} \mu_S' &= \boldsymbol{m}_S(\boldsymbol{x}') + k_S(\boldsymbol{x}', X_S) C_S(X_S, X_S)^{-1} (\boldsymbol{y}_S - \boldsymbol{m}_S(X_S)) \\ C_S' &= k_S(\boldsymbol{x}', \boldsymbol{x}') + \sigma_{nS}^2 - k_S(\boldsymbol{x}', X_S) C_S(X_S, X_S)^{-1} k_S(X_S, \boldsymbol{x}') \end{aligned} \tag{29}$$

2) The second configuration is no retraining N: Although only the source data are used for learning the hyperparameters during training, the prediction combines these learned hyperparameters with the source and target data $X_D = \{X_T, X_S\}$. This configuration represents the scenario where data collected after training are added to the training set under the assumption that the underlying distribution is unchanged. Comparing this configuration with FAT-GP will help quantify the value of knowing the task assignment for the datasets:

Hyperparameters:

$$\theta_S = \{\sigma_{fS}^2, \sigma_{nS}^2, L_S, \boldsymbol{m}_S\}$$

Training:

$$\theta_S^* = \arg\max_{\theta_S} \ell_n \; p(\boldsymbol{y}_S | X_S, \theta_S)$$

Prediction:

$$\begin{aligned} \mu_N' &= \boldsymbol{m}_S(\boldsymbol{x}') + k_S(\boldsymbol{x}', X_D) C_S(X_D, X_D)^{-1} (\boldsymbol{y}_D - \boldsymbol{m}_S(X_D)) \\ C_N' &= k_S(\boldsymbol{x}', \boldsymbol{x}') + \sigma_{nS}^2 - k_S(\boldsymbol{x}', X_D) C_S(X_D, X_D)^{-1} k_S(X_D, \boldsymbol{x}') \end{aligned} \tag{30}$$

3) The third configuration is all data training D: As mentioned in Sec. III.A, this configuration ignores separation between X_T and X_S, and it uses $X_D = [X_T^T, X_S^T]^T$ for learning hyperparameters. Like the no-retraining R configuration, this also helps quantify the value of knowing the task assignment, but this incurs an additional training overhead:

Hyperparameters:

$$\theta_D = \{\sigma_f^2, \sigma_n^2, L, \boldsymbol{m}_D\} \tag{31}$$

Training:

$$\theta_D^* = \arg\max_{\theta_D} \ell_n \; p(\boldsymbol{y}_D | X_D, \theta_D)$$

Prediction:

$$\begin{aligned} \mu_D' &= \boldsymbol{m}_D(\boldsymbol{x}') + k_D(\boldsymbol{x}', X_D) C_D(X_D, X_D)^{-1} (\boldsymbol{y}_D - \boldsymbol{m}_D(X_D)) \\ C_D' &= k_D(\boldsymbol{x}', \boldsymbol{x}') + \sigma_{nD}^2 - k_D(\boldsymbol{x}', X_D) C_D(X_D, X_D)^{-1} k_D(X_D, \boldsymbol{x}') \end{aligned}$$

The learning configurations were compared by learning models for 1-D examples with 1) a shift between tasks such that $y_T = y_S + 2$, and 2) a scaling $y_T = 2y_S$ (i.e., Fig. 2). Each learner was provided with the same data: source GP, 65 source training samples, and 10 target training samples. The experiment was repeated 20 times. Table 2 presents the RMSE and MSLL for each of the learning configurations, averaged over 20 runs. Because the source GP makes no use of target data, the MSLL values were calculated with the source GP as the baseline. All other configurations were expected to improve over this baseline, i.e., have negative MSLL values.

Table 2 FAT-GP vs other GP configurations

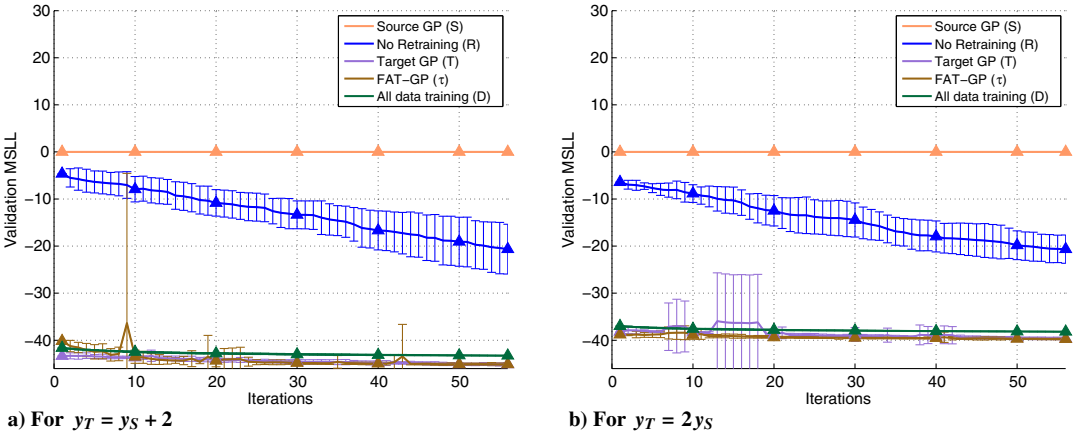
	$y_T = y_S + 2$		$y_T = 2y_S$	
Learning configuration	RMSE	MSLL	RMSE	MSLL
Source GP S	2.06	0	1.83	0
No retraining N	1.99	−4.62	1.75	−6.41
All data training D	1.95	−41.62	1.78	−36.97
Target GP T	0.75	−43.32	1.42	−38.18
FAT-GP τ	0.59	−40.16	0.57	−38.79

Examining the MSLL values in the table shows that ignoring target data, as in the source GP case, results in the worst performance. Although the no-retraining *N* case improved over the source GP, models that learned hyperparameters from the target data did significantly better than the *S* and *N* configurations. Although configurations *T*, τ, and *D* had comparable MSLL values, FAT-GP reduced the RMSE over the other learners.

B. Effect of Target Data Size on Transfer

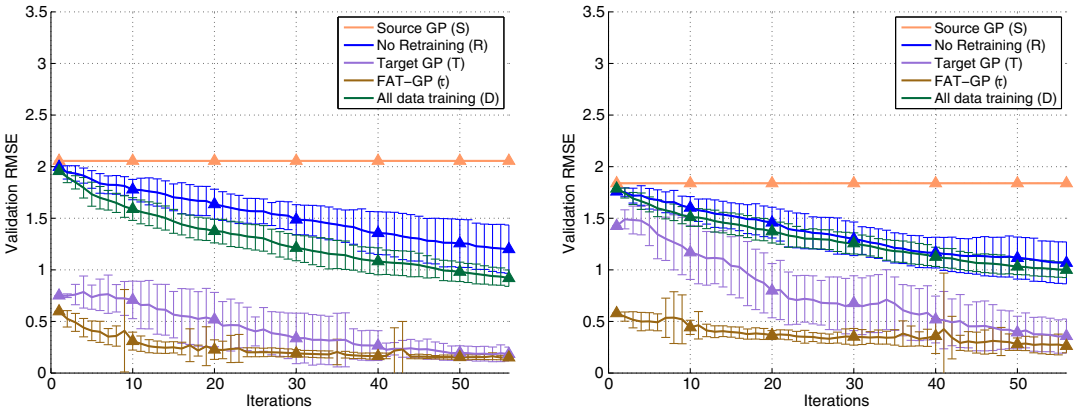
To examine the effect of increasing target training data, the FAT-GP and the other learners were trained iteratively. Starting with 10 training samples from the target task at iteration 1, a new random target training sample was added at each iteration. This was repeated for 56 iterations until both the source and target tasks had 65 measurements. The performance of the learners at each iteration was evaluated against a fixed validation set.

Figure 12 shows the MSLL and RMSE results of the five learners over 56 iterations. Increasing the target data has no impact on the source GP, which shows up as a flat line in both plots. In the case of configuration *N*, as the proportion of the target data goes up, it begins to capture the statistics of the target task, gradually improving its MSLL. However, as in Sec. VII.A, configurations *T*, τ, and *D* outperform *S* and *N* in terms of the MSLL. Even so, Fig. 13 shows that the RMSE of learner *D* is close to that of *N*, with the target GP and FAT-GP achieving a much lower RMSE by the last iteration.



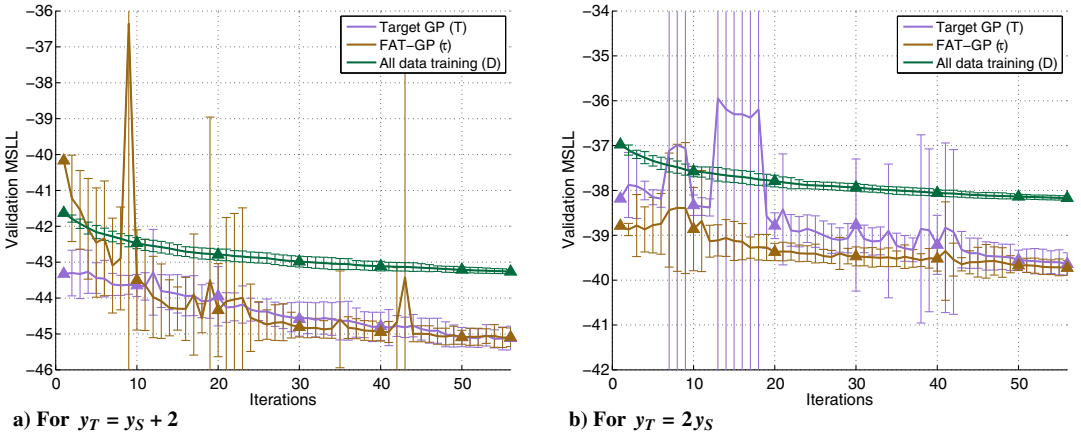
a) For $y_T = y_S + 2$ b) For $y_T = 2y_S$

Fig. 12 Effect of target data size on the MSLL for FAT-GP and other learning configurations.



a) For $y_T = y_S + 2$ b) For $y_T = 2y_S$

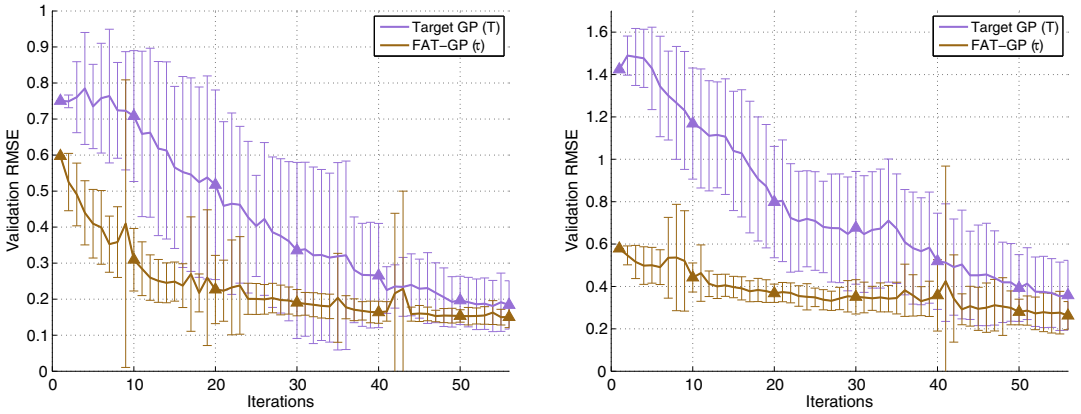
Fig. 13 Effect of target data size on the RMSE for FAT-GP and other learning configurations.



a) For $y_T = y_S + 2$ b) For $y_T = 2y_S$

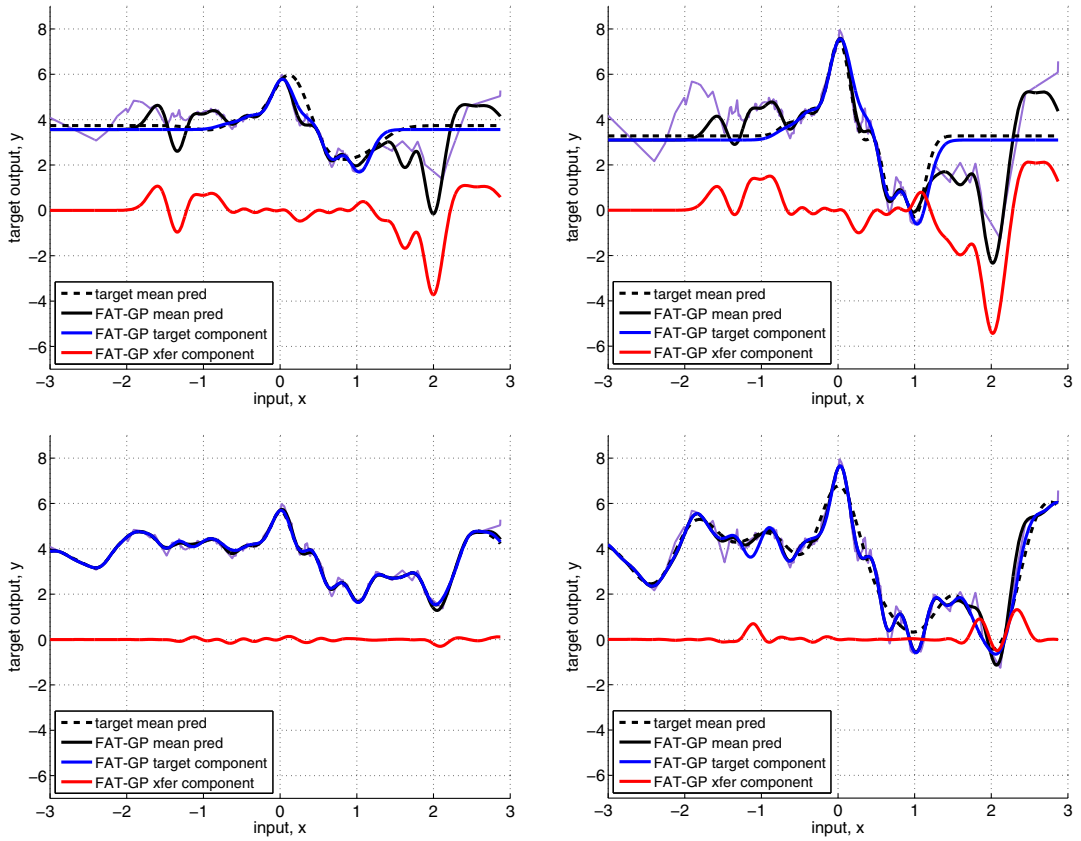
Fig. 14 FAT-GP vs target GP MSLL comparison.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437



a) For $y_T = y_S + 2$ b) For $y_T = 2y_S$

Fig. 15 FAT-GP vs target GP RMSE comparison.



a) For $y_T = y_S + 2$ b) For $y_T = 2y_S$

Fig. 16 Evolution of FAT-GP's mean components from iteration 1 with 10 target samples (shown in top row) to iteration 56 with 65 samples (shown in bottom row).

Figures 14 and 15 take a closer look at how the target GP and FAT-GP compare. Although the FAT-GP's MSLL is marginally better than the target GP, Fig. 15 shows that the FAT-GP has a lower RMSE compared to the target GP. In fact, in both examples, the target GP RMSE at the last iteration is reached by FAT-GP around iteration 20. This highlights how FAT-GP can provide an efficient interim model while more target task data are being collected.

As the target task's training set grows beyond 40, the big divide between the RMSE performance of the target and FAT-GP begins to narrow. As the information in the target training set increases, the need to transfer knowledge decreases. This is reflected in Fig. 16, which shows change in the mean components (discussed in Sec. V.A) between the first and last iterations. The top row reproduces the plots from Fig. 8 for comparison with those in the bottom row, which are from the last iteration. Compared to iteration 1, the target GP (dashed line), FAT-GP (darkest solid line), and FAT-GP's target component all closely match the (light solid line) true target, whereas the (lowest line) transfer component is mostly a flat line. On the lines of the SNR discussion in Sec. V.A, as the noise in the data goes down and the SNR grows, the contribution of transfer greatly diminishes.

VI. Conclusions

This paper describes a forward adaptive transfer learning method, FAT-GP, which allows robots to leverage previously learned Gaussian process regression models and use them as sources of information in new learning tasks. This is especially valuable when limited training data are available for the new target task. FAT-GP decouples the kernel and hyperparameter selection for the target task from those of the source task,

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

providing an inference framework that is desirable when dealing with real-world dynamic environments. Additionally, because the source task's large covariance matrix is precomputed, FAT-GP amortizes cost and is computationally cheaper than other GP approaches using transfer kernels. Simulations studies on 1-D and two-dimensional examples show that similar source tasks can considerably improve the target's performance. More important, the FAT-GP exploits the correlations between the source and target to achieve a low error with much less target data, and can thus serve as an efficient model in the interim as more target data are collected through exploration.

Appendix A: Gaussian Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. The details of the Gaussian process framework can be found in [1,2,40], and the main points are summarized in this appendix.

Once the GP is trained, the learned model can be used to predict the output y' at unseen state $\boldsymbol{x}'$. This prediction is in the form of a probability density function (PDF), which comprises the expected value (mean) and the variance:

$$p(y') = \mathcal{N}(\mu_y(\boldsymbol{x}'), \sigma_y^2(\boldsymbol{x}')) \triangleq \mathcal{GP}(\boldsymbol{x}'|\{\boldsymbol{x}_{1:N}, \boldsymbol{y}_{1:N}, \theta\}) \tag{A1}$$

where θ are the hyperparameters of the Gaussian process, and $\{\boldsymbol{x}_{1:N}, \boldsymbol{y}_{1:N}, \theta\}$ is the training dataset of N input–output pairs. This PDF is obtained by calculating the joint distribution of the unseen state with the states of the training samples as follows:

$$\begin{aligned}\mu_y(\boldsymbol{x}\prime) &= k(\boldsymbol{x}, \boldsymbol{x}')^T (K + \sigma_n^2 I)^{-1} \boldsymbol{y} \\ \sigma_y^2(\boldsymbol{x}') &= k(\boldsymbol{x}', \boldsymbol{x}') - k(\boldsymbol{x}, \boldsymbol{x}')^T (K + \sigma_n^2 I)^{-1} k(\boldsymbol{x}, \boldsymbol{x}')\end{aligned} \tag{A2}$$

Here, $k(\boldsymbol{x}, \boldsymbol{x}')$ is the $N \times 1$ vector of correlations of the new state with all the training points' states, $\boldsymbol{y}$ is the $N \times 1$ vector of measured variations at the training points, K is an $N \times N$ kernel matrix with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and σ_n^2 is assumed to be the noise variance of the original process.

The GP computes correlations using a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$. The choice of the correlation function or kernel is a key design decision when using a GP. Due to its infinite differentiability, a squared exponential or Gaussian kernel is popularly used. It is defined as

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T L^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)\right) \tag{A3}$$

where σ_f^2 is the signal variance, and L is a $D \times D$ diagonal matrix when $\boldsymbol{x} \in \mathbb{R}^D$. The diagonal elements are given by $L(d, d) = l_d^2$, where l_d is the length scale of dimension d, where $d = 1, \ldots, D$. Furthermore, it is common to assume a spatial isotropic GP in which there is a single length scale for all dimensions, e.g., $L = l_s * I$, where I is the identity matrix.

Training a Gaussian process involves learning σ_n, σ_f, and L, which are the hyperparameters θ of the model. The hyperparameters are derived by maximizing the log likelihood function of the Gaussian process for the n sample points in the training dataset:

$$\begin{aligned}\theta &= \arg\min_{\theta} \ln p(\boldsymbol{y}|\theta) \\ &= \arg\min_{\theta}\left[-\frac{1}{2}\boldsymbol{y}^T C_N^{-1} \boldsymbol{y} - \frac{1}{2}\ln|C_N| - \frac{N}{2}\ln 2\pi\right]\end{aligned} \tag{A4}$$

where $C_N(\theta) = K(\{\boldsymbol{x}\}_{1:n}; \sigma_f, L) + \sigma_n^2 I$.

Appendix B: FAT-GP Mean Components

The mean prediction equation for the FAT-GP is given by

$$\begin{aligned}\mu_\tau' &= \boldsymbol{m}_T(\boldsymbol{x}') + k_\tau(\boldsymbol{x}', X_D) C_\tau(X_D, X_D)^{-1} (\boldsymbol{y}_D - \boldsymbol{m}_\tau(X_D)) \\ &= \boldsymbol{m}_T(\boldsymbol{x}') + [k_\tau(\boldsymbol{x}', X_T) \quad k_\tau(\boldsymbol{x}', X_S)] \begin{bmatrix} C_{TT} & K_{TS} \\ K_{ST} & K_{SS} \end{bmatrix}^{-1} \left(\begin{bmatrix} \boldsymbol{y}_T \\ \boldsymbol{y}_S \end{bmatrix} - \begin{bmatrix} \boldsymbol{m}_T(X_T) \\ \boldsymbol{m}_S(X_S) \end{bmatrix} \right)\end{aligned}$$

Using the formula for inverses of block matrices, we get

$$\begin{aligned}C_\tau(X_D, X_D)^{-1} &= \begin{bmatrix} C_{TT} & K_{TS} \\ K_{ST} & K_{SS} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} C_{TT}^{-1} + C_{TT}^{-1} K_{TS} M K_{ST} C_{TT}^{-1} & -C_{TT}^{-1} K_{TS} M \\ -K_{ST} C_{TT}^{-1} & M \end{bmatrix}\end{aligned}$$

where $M = [C_{SS} - K_{ST} C_{TT}^{-1} K_{TS}]^{-1}$. Notice that

$$M = C_{S|T}^{-1} = [C_{SS} - K_{ST} C_{TT}^{-1} K_{TS}]^{-1}$$

which is given by

$$p(\boldsymbol{y}_S|\boldsymbol{y}_T, X_T, X_S, \theta_T, \theta_S) = \mathcal{N}(\mu_{S|T}, C_{S|T})$$

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

where $\mu_{S|T} = \boldsymbol{m}_S + K_{ST}C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T)$:

$$C_{S|T} = C_{SS} - K_{ST}C_{TT}^{-1}K_{TS}$$

Plugging the inverse back into the mean prediction expression, we get

$$\mu'_\tau = \boldsymbol{m}_T(\boldsymbol{x}') + \begin{bmatrix} k_\tau(\boldsymbol{x}', X_T) & k_\tau(\boldsymbol{x}', X_S) \end{bmatrix} \begin{bmatrix} C_{TT}^{-1} + C_{TT}^{-1}K_{TS}C_{S|T}^{-1}K_{ST}C_{TT}^{-1} & -C_{TT}^{-1}K_{TS}C_{S|T}^{-1} \\ -C_{S|T}^{-1}K_{ST}C_{TT}^{-1} & C_{S|T}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_T - \boldsymbol{m}_T(X_T) \\ \boldsymbol{y}_S - \boldsymbol{m}_S(X_S) \end{bmatrix}$$

and expanding the expression, we get

$$\begin{aligned}\mu'_\tau = {} & \boldsymbol{m}_T(\boldsymbol{x}') + k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) + k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}K_{TS}C_{S|T}^{-1}K_{ST}C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) - k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}K_{TS}C_{S|T}^{-1}(\boldsymbol{y}_S - \boldsymbol{m}_S(X_S)) \\ & - k_S(\boldsymbol{x}', X_S)C_{S|T}^{-1}K_{ST}C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) + k_S(\boldsymbol{x}', X_S)C_{S|T}^{-1}(\boldsymbol{y}_S - \boldsymbol{m}_S(X_S))\end{aligned}$$

Grouping the terms with common factors,

$$\begin{aligned}\mu'_\tau = {} & \boldsymbol{m}_T(\boldsymbol{x}') + k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) + [k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}K_{TS} - k_S(\boldsymbol{x}', X_S)] \times [C_{SS} - K_{ST}C_{TT}^{-1}K_{TS}]^{-1} \\ & \times [K_{ST}C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) - (\boldsymbol{y}_S - \boldsymbol{m}_S(X_S))]\end{aligned}$$

This equation can also be written in terms of cross-covariance matrices and the similarity measure as

$$\begin{aligned}\mu'_\tau = {} & \boldsymbol{m}_T(\boldsymbol{x}') + k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) + [\lambda k_T(\boldsymbol{x}', X_T)C_{TT}^{-1}K_{\text{cross}}(X_T, X_S) - k_S(\boldsymbol{x}', X_S)] \\ & \times [C_{SS} - \lambda^2 K_{\text{cross}}(X_S, X_T)C_{TT}^{-1}K_{\text{cross}}(X_T, X_S)]^{-1} \times [\lambda K_{\text{cross}}(X_S, X_T)C_{TT}^{-1}(\boldsymbol{y}_T - \boldsymbol{m}_T(X_T)) - (\boldsymbol{y}_S - \boldsymbol{m}_S(X_S))]\end{aligned}$$

Appendix C: Impact of Signal-to-Noise Ratio on Variance Components

Equation (22) shows that the variance prediction of a FAT-GP contains two terms that reduce the uncertainty.

Figure 10 shows that the reduction due to transfer goes to zero when a target sample is colocated with a source sample. This section analyzes the root of this interesting behavior.

To understand how training samples affect variance predictions, we examine the interactions between individual samples with an unseen input sample $\boldsymbol{x}'$. The correlations of $\boldsymbol{x}'$ with a source task sample $\boldsymbol{x}_S$ and a target task sample $\boldsymbol{x}_T$, for a given set of source and FAT-GP hyperparameters, are given by

$$\begin{aligned}k_\tau(\boldsymbol{x}', \boldsymbol{x}_T) &= c_{\tau(T)}\sigma_{f\tau(T)}^2 \\ k_\tau(\boldsymbol{x}', \boldsymbol{x}_S) &= \lambda c_S \sqrt{\sigma_{f\tau(T)}^2 \sigma_{fS}^2} \sqrt{\frac{2l_{\tau(T)}l_S}{l_{\tau(T)}^2 + l_S^2}}\end{aligned}$$

where

$$\begin{aligned}c_S &= \exp\left[-\frac{\|\boldsymbol{x}' - \boldsymbol{x}_S\|^2}{l_{\tau(T)}^2 + l_S^2}\right] \\ c_{\tau(T)} &= \exp\left[-\frac{\|\boldsymbol{x}_T - \boldsymbol{x}'\|^2}{2l_{\tau(T)}^2}\right]\end{aligned}$$

Their correlation of the source and target samples can also be calculated as

$$\begin{aligned}k(\boldsymbol{x}_T, \boldsymbol{x}_S) &= \lambda \sqrt{\sigma_{f\tau(T)}^2 \sigma_{fS}^2} \sqrt{\frac{2l_{\tau(T)}l_S}{l_{\tau(T)}^2 + l_S^2}} \exp\left[-\frac{\|\boldsymbol{x}_T - \boldsymbol{x}_S\|^2}{l_{\tau(T)}^2 + l_S^2}\right] \\ &= \lambda c_{TS} \sqrt{\sigma_{f\tau(T)}^2 \sigma_{fS}^2} \sqrt{\frac{2l_{\tau(T)}l_S}{l_{\tau(T)}^2 + l_S^2}}\end{aligned}$$

where

$$c_{TS} = \exp\left[-\frac{\|\boldsymbol{x}_T - \boldsymbol{x}_S\|^2}{l_{\tau(T)}^2 + l_S^2}\right]$$

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.1010437

Thus, the reduction due to transfer [last term in Eq. (22)] is written as

$$K_{ST}C_{TT}^{-1}k_{\tau}(X_T,\boldsymbol{x}') - k_{\tau}(X_S,\boldsymbol{x}') = \lambda b\sqrt{\sigma^2_{f\tau(T)}\sigma^2_{fS}}\sqrt{\frac{2l_{\tau(T)}l_S}{l^2_{\tau(T)}+l^2_S}}$$

where b is a coefficient of transfer and is given by

$$b = \frac{c_{TS}c_{\tau(T)}\sigma^2_{f\tau(T)} - c_S\sigma^2_{f\tau(T)} - c_S\sigma^2_{n\tau(T)}}{\sigma^2_{f\tau(T)}+\sigma^2_{n\tau(T)}}$$

Depending on the relative positions of $\boldsymbol{x}_S, \boldsymbol{x}_T$, and $\boldsymbol{x}'$, the coefficient of transfer will differ. When the source and the target sample are colocated with $\boldsymbol{x}'$, $c_{TS} = c_S = c_T = 1$. Then, the coefficient of transfer is given by

$$b = -\frac{\sigma^2_{n\tau(T)}}{\sigma^2_{f\tau(T)}+\sigma^2_{n\tau(T)}}$$

When $\boldsymbol{x}'$ is outside the source's distance of correlation (i.e., $c_S = 0$), it influences $\boldsymbol{x}'$ via the target samples close to it. Here, the coefficient of transfer is given by

$$b = \frac{c_{TS}c_{\tau(T)}\sigma^2_{f\tau(T)}}{\sigma^2_{n\tau(T)}+\sigma^2_{f\tau(T)}}$$

When $\boldsymbol{x}'$ is outside the distance of correlation of all target samples (i.e., $c_{\tau(T)} = 0$) but is close to a source sample, the source will impact $\boldsymbol{x}'$ proportional to

$$b = -c_S$$

Thus, in the absence of target samples in the region of $\boldsymbol{x}'$, the variance prediction depends directly on the correlation between $\boldsymbol{x}_S$ and $\boldsymbol{x}'$.

References

[1] Rasmussen, C. E., and Williams, C. K. I., *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.

[2] Seeger, M., "Gaussian Processes for Machine Learning," *International Journal of Neural Systems*, Vol. 14, No. 2, 2004, pp. 69–106. doi:10.1142/S0129065704001899

[3] Guestrin, C., Krause, A., and Singh, A. P., "Near-Optimal Sensor Placements in Gaussian Processes," *Proceedings of the 22nd International Conference on Machine Learning–ICML '05*, ACM Press, New York, Aug. 2005, pp. 265–272.

[4] Krause, A., and Guestrin, C., "Nonmyopic Active Learning of Gaussian Processes: An Exploration Exploitation Approach," *Proceedings of the 24th International Conference on Machine Learning (ICML)*, ACM Press, New York, 2007, pp. 449–456.

[5] Plagemann, C., Kersting, K., and Burgard, W., "Nonstationary Gaussian Process Regression Using Point Estimates of Local Smoothness," *Machine Learning and Knowledge Discovery in Databases*, Vol. 5212, Lecture Notes in Computer Science, Springer, New York, 2008, pp. 204–219.

[6] Hollinger, G. A., Pereira, A. A., and Sukhatme, G. S., "Learning Uncertainty Models for Reliable Operation of Autonomous Underwater Vehicles," *International Conference on Robotics and Automation*, IEEE Publ., Piscataway, NJ, 2013, pp. 5593–5599.

[7] Ko, J., and Fox, D., "GP-BayesFilters: Bayesian Filtering Using Gaussian Process Prediction and Observation Models," *International Conference on Intelligent Robots and Systems*, Vol. 27, IEEE Publ., Piscataway, NJ, May 2008, pp. 75–90.

[8] Grande, R. C., Chowdhary, G., and How, J. P., "Experimental Validation of Bayesian Nonparametric Adaptive Control Using Gaussian Processes," *Journal of Aerospace Information Systems*, Vol. 11, No. 9, 2014, pp. 565–578. doi:10.2514/1.I010190

[9] Chowdhary, G., Kingravi, H. A., How, J. P., and Vela, P. A., "Bayesian Nonparametric Adaptive Control Using Gaussian Processes," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 26, No. 3, 2015, pp. 537–550. doi:10.1109/TNNLS.2014.2319052

[10] Hardy, J., Havlak, F., and Campbell, M., "Multiple-Step Prediction Using a Two Stage Gaussian Process Model," *2014 American Control Conference*, IEEE Publ., Piscataway, NJ, June 2014, pp. 3443–3449.

[11] Fink, J., and Kumar, V., "Online Methods for Radio Signal Mapping with Mobile Robots," *2010 IEEE International Conference on Robotics and Automation*, IEEE, Piscataway, NJ, May 2010, pp. 1940–1945.

[12] Wagle, N., and Frew, E., "Spatio-Temporal Characterization of Airborne Radio Frequency Environments," *Wireless Networking for Unmanned Autonomous Vehicles*, IEEE, Piscataway, NJ, Dec. 2011, pp. 1269–1273.

[13] Lawrance, N., and Sukkarieh, S., "Path Planning for Autonomous Soaring Flight in Dynamic Wind Fields," *2011 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ, 2011, pp. 2499–2505.

[14] Plonski, P. A., Tokekar, P., and Isler, V., "Energy-Efficient Path Planning for Solar-Powered Mobile Robots," *Journal of Field Robotics*, Vol. 30, No. 4, July 2013, pp. 583–601. doi:10.1002/rob.2013.30.issue-4

[15] Csató, L., and Opper, M., "Sparse On-Line Gaussian Processes," *Neural Computation*, Vol. 14, No. 3, 2002, pp. 641–668. doi:10.1162/089976602317250933

[16] Quiñonero-Candela, J., Rasmussen, C. E., and Herbrich, R., "A Unifying View of Sparse Approximate Gaussian Process Regression," *Journal of Machine Learning Research*, Vol. 6, Dec. 2005, pp. 1935–1959.

[17] Wagle, N., and Frew, E. W., "Transfer Learning for Dynamic RF Environments," *Proceedings of the American Control Conference*, IEEE Publ., Piscataway, NJ, 2012, pp. 1406–1411.

[18] Arnold, A., Nallapati, R., and Cohen, W. W., "A Comparative Study of Methods for Transductive Transfer Learning," *7th IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, IEEE, Piscataway, NJ, Oct. 2007, pp. 77–82.

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437

[19] Pan, S. J., and Yang, Q., "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, 2010, pp. 1345–1359. doi:10.1109/TKDE.2009.191
[20] Tekdas, O., Yang, W., and Isler, V., "Robotic Routers: Algorithms and Implementation," *International Journal of Robotics Research*, Vol. 29, No. 1, 2010, pp. 110–126. doi:10.1177/0278364909105053
[21] Ghaffarkhah, A., and Mostofi, Y., "Communication-Aware Motion Planning in Mobile Networks," *IEEE Transactions on Automatic Control*, Vol. 56, No. 10, 2011, pp. 2478–2485. doi:10.1109/TAC.2011.2164033
[22] Stachura, M., and Frew, E. W., "Cooperative Target Localization with a Communication Aware Unmanned Aircraft System," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 5, Sept. 2011, pp. 1352–1362. doi:10.2514/1.51591
[23] Vieira, M. A., Taylor, M. E., Tandon, P., Jain, M., Govindan, R., Sukhatme, G. S., and Tambe, M., "Mitigating Multi-Path Fading in a Mobile Mesh Network," *Ad Hoc Networks*, Vol. 11, No. 4, Feb. 2011, pp. 1510–1521.
[24] Carfang, A. J., Wagle, N., and Frew, E. W., "Improving Data Ferrying by Iteratively Learning the Radio Frequency Environment," *Journal of Aerospace Information Systems*, 2015, (accepted).
[25] Stachura, M., and Frew, E., "Communication-Aware Information Gathering Experiments with an Unmanned Aircraft System," *Journal of Field Robotics*, Jan. 2017. doi:10.1002/rob.21666
[26] Malmirchegini, M., and Mostofi, Y., "On the Spatial Predictability of Communication Channels," *IEEE Transactions on Wireless Communications*, Vol. 11, No. 3, 2012, pp. 964–978. doi:10.1109/TWC.2012.012712.101835
[27] Phillips, C., Sicker, D., and Grunwald, D., "A Survey of Wireless Path Loss Prediction and Coverage Mapping Methods," *IEEE Communications Surveys and Tutorials*, Vol. 15, No. 1, 2013, pp. 255–270. doi:10.1109/SURV.2012.022412.00172
[28] Wagle, N., and Frew, E. W., "Online Evaluation of Communication Models Derived via Transfer Learning," *Globecom Workshop on Wireless Networking or Unmanned Autonomous Vehicles*, IEEE Publ., Piscataway, NJ, Dec. 2012, pp. 1609–1613.
[29] Minka, T. P., and Picard, R. W., "Learning How to Learn Is Learning with Point Sets," MIT Media Lab., MIT TR, Cambridge, MA, 1997.
[30] Lawrence, N. D., and Platt, J. C., "Learning to Learn with the Informative Vector Machine," *Proceedings of the 21st International Conference on Machine Learning*, ACM Press, New York, 2004, p. 65.
[31] Schwaighofer, A., Tresp, V., and Yu, K., "Learning Gaussian Process Kernels Via Hierarchical Bayes," *Advances in Neural Information Processing Systems*, Vol. 17, 2005, pp. 1209–1216.
[32] Chai, K. M. A., "Generalization Errors and Learning Curves for Regression with Multi-Task Gaussian Processes," *Advances in Neural Information Processing Systems*, Vol. 22, 2009, pp. 279–287.
[33] Cao, B., Pan, S. J., Zhang, Y., Yeung, D.-Y., and Yang, Q., "Adaptive Transfer Learning," *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, AAAI, Palo Alto, CA, 2010, pp. 407–412.
[34] Kulis, B., Saenko, K., and Darrell, T., "What You Saw Is Not What You Get: Domain Adaptation Using Asymmetric Kernel Transforms," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ, 2011, pp. 1785–1792.
[35] Hoffman, J., Rodner, E., Donahue, J., Darrell, T., and Saenko, K., "Efficient Learning of Domain-Invariant Image Representations," *International Conference on Learning Representations (ICLR)*, 2013, pp. 1–9.
[36] Kandemir, M., "Asymmetric Transfer Learning with Deep Gaussian Processes," *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, JMLR, Brookline, MA, 2015, pp. 730–738.
[37] Leen, G., Peltonen, J., and Kaski, S., "Focused Multi-Task Learning in a Gaussian Process Framework," *Machine Learning*, Vol. 89, Nos. 1–2, 2012, pp. 157–182.
[38] Melkumyan, A., and Ramos, F., "Multi-Kernel Gaussian Processes," *International Joint Conference on Artificial Intelligence*, AAAI, Palo Alto, CA, 2011, pp. 1408–1413.
[39] Paciorek, C. J., and Schervish, M. J., "Nonstationary Covariance Functions for Gaussian Process Regression," *Advances in Neural Information Processing Systems*, Vol. 16, 2004, pp. 273–280.
[40] Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, New York, 2009, pp. 303–320.

J. P. How
Associate Editor

Downloaded by UNIVERSITY OF COLORADO BOULDER on July 13, 2017 | http://arc.aiaa.org | DOI: 10.2514/1.I010437